Winter 1993

Physical behavior of computing networks with an emphasis on routing

Peter George Drexel
University of New Hampshire, Durham

Order Number 9420563

Physical behavior of computing networks with an emphasis on routing

Drexel, Peter George, Ph.D.

University of New Hampshire, 1993

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

PHYSICAL BEHAVIOR OF COMPUTING NETWORKS WITH AN EMPHASIS ON ROUTING

BY

Peter George Drexel
B.S., Rochester Institute of Technology, 1969
M.S., Rochester Institute of Technology, 1980

DISSERTATION

Submitted to the University of New Hampshire
in Partial Fulfillment of
the Requirements for the Degree of

Doctor of Philosophy

in

Engineering

December, 1993

This dissertation has been examined and approved.

Andrzej Rucinski

Dissertation Director, Andrzej Rucinski
Associate Professor of Electrical and Computer Engineering

R. Daniel Bergeron, Professor of Computer Science

Loren D. Meeker, Professor of Mathematics

John Pokoski, Professor of
Electrical and Computer Engineering

Charles Taft, Professor Emeritus
of Mechanical Engineering

September 22, 1993
Date

DEDICATION

This work is dedicated to my father: Georg Andreas Drechsel.
Your mountain climbing inspired this achievement.

ACKNOWLEDGEMENTS

I wish to thank my advisor Andrzej Rucinski for his insight, guidance and encouragement. Without his help this research would not have been possible.

I wish to thank Dan Bergeron, David Meeker, John Pokoski and Charles Taft for serving on my guidance committee. I also wish to thank David Limbert for his support in completing this process.

Moreover, I wish to thank my colleagues: Larry Blaine and Zhizhang Shen for their collaboration and encouragement. Their efforts have made significant contributions to the research. In addition, I wish to thank the students who assisted in this work: Rose Bente, Jerry Munoz, Liam Urbach and Catherine Wilson.

Lastly, I want to thank my family, especially my wife Kathy, for their patience, encouragement and understanding throughout the Ph.D. process.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

PHYSICAL BEHAVIOR OF COMPUTING NETWORKS WITH AN EMPHASIS ON ROUTING

BY

Peter George Drexel
University of New Hampshire, December 1993

Combining powerful, local computing capability with inexpensive packet switched communication invites the creation of a new entity - the computing network. A computing network is a group of computer systems, interconnected by a packet switched network, which cooperate to accomplish a specific task.

Efficient communication between computer systems is critical. The accepted method of dealing with packet congestion is to limit network operation to linear regions (≤ 80 percent of total capacity).

Because of large parameter spaces, the behavior of computing networks is complex and not easily predictable with current techniques except in the simplest, linear region of operation. Since traditional approaches are inadequate, examining computing networks from a different perspective is desirable.

The complex behavior of a computing network in some sense emulates the behavior of many natural systems. The computing network may be thought of as a "pseudo-physical system" (PPS). A PPS is any complex, artificial system which exhibits a critical subset of the characteristics of a natural system such as the presence of a fractal dimension combined with a "rich" spectral signature.

Natural systems operate close to their functional capacity. Combining the ideas of computing networks and pseudo-physical systems creates a new paradigm. This new paradigm captures the behavior of computing networks for the purpose of improved communication and processing capability.

Extensive research, comparing distributed computing systems to natural ecologies, alludes to the link between order in systems and "chaos." In many natural systems, chaos implies order.

The major contribution of this research is the "discovery" of chaos in computing networks by means of an innovative technique: the combination of the Correlation Dimension and the Fast Fourier Transform. The implication of the research is it opens a new area of study which should lead to "physically-friendly" computing networks. The conclusion of this research is that the measurement of chaos in computing networks implies order (yet unused).

CHAPTER I

INTRODUCTION

The combination of powerful, local computing capability with inexpensive packet switched communication invites the creation of a new entity - the computing network. A computing network is defined as a group of computer systems, interconnected by a packet switched network, which cooperate to accomplish a specific task.

Because of their large parameter space, the behavior of computing networks is complex and not easily predictable with current techniques except in the simplest, linear region of operation. The inadequacies of traditional approaches necessitate a new perspective, which is the focus of this research.

1.1 Computing Networks

Convenient access to packet switched networks has increasingly become an expected part of computer systems [Halsall, 1992], [Martin, 1990]. The spectrum of these interconnecting networks ranges from wide area network (WAN) and local area network (LAN) [Stallings, 1993] to closely coupled multi-computer systems [Patterson, 1993]. It is probable that packet switched networks will eventually connect computer to computer, planet-wide [Kobb, 1993].

In packet-switched networks, data is broken into groups

of binary bits called packets. Therefore, packets are subsets of a message. Packets contain data, control and error detection information. For a description of these concepts, see [Fitzgerald, 1993].

Computationally intensive applications, such as image processing [El Mesbahi, 1990], machine vision [Weems, 1992] and computational physics [Boghosian, 1990], currently run on massively parallel computer systems (MPCS) [Stone, 1990], [Gibson, 1991]. Because of the economical nature of modern microcomputer technology, it is likely that extremely powerful computers will eventually replace the desktop PC. Packet switched networks will allow these systems to be linked to form MPCS-like entities: computing networks. The advantages of utilizing this computing power to tackle complex tasks are obvious.

1.2 Bandwidth Problem in Systems

Efficient inter-processor communication is essential for cooperative behavior to exist in the computing network. Similar to the dialogue between members of a research team, inter-processor communication exists on a local (nearest neighbors) and a global scale (task initiation, management and completion). Close cooperation between processors requires significant packet traffic. A fundamental problem with distributed applications is the delay encountered when sending messages from processor to processor. In [Patterson, 1993] the authors indicate that this delay is so large as to obscure

the underlying multi-processor topology. Thus, efficient use of the communication link between processors is critical.

When the network's total processor population interacts to complete a task, collisions result. In other words, if two processors send packets at the same time and in the same communication space, these packets collide. Not only do packet collisions cause loss of data but retransmission of lost packets adds to network load. If the load on the computing network exceeds a critical limit, packet collisions increase dramatically until the network "freezes" (i.e., no packets are successfully transferred from processor to processor and deadlock occurs). In [Stallings, 1991], this load-delay congestion problem is characterized as a limiting factor in network capability.

At present, the commonly accepted method of dealing with packet congestion is to limit network operation to linear regions. "Linear region" is used here to mean that the change in the offered load of the network is equal to the increase in the packet traffic. Typically, this operation is $\leq$ 80 percent of total network capacity [Stallings, 1991]. At present, the remaining 20 percent of system capacity is unused. Congestion control means limiting the amount of traffic that may enter the system: either by buffering the traffic enroute or by destroying packets when delay reaches intolerable levels. An alternative is to upgrade the network to improve its bandwidth. Neither approach solves the problem. All are expensive and inefficient because they deal with symptoms

rather than cause.

1.3 Computing Networks as Natural Systems

[Bertsekas, 1992] and [Schwartz, 1988] provide good examples of the standard approach to the analysis of networks. These authors introduce queuing theory, develop models for various network elements and estimate network behavior based on the models. A problem with the standard approach is that the representation of the network, for a network of any appreciable size, is cumbersome. Moreover, because these models use approximations, comparisons between such models and the real systems they represent are not always satisfactory (i.e., the model does not effectively capture the nature of the complex system). Relative to the analysis of congestion in networks, Price states: "This kind of study could certainly not be carried out by means of queuing theory and mathematical analysis; these methods demand simplifying assumptions which would obscure results being sought [Price, 1978]."

The complex behavior of a computing network, in the performance of a specific task, in some sense emulates the behavior of a natural system. In [Seeley, 1989] the concept of a super-organism is introduced. A super-organism collectively represents the behavior of a group of cooperating sub-organisms. In this case, the sub-organisms are honey bees and the super-organism is their hive.

HYPOTHESIS 1: Computing networks may be viewed as "pseudo-physical systems" (PPS).

A pseudo-physical system is defined as any complex, artificial system which exhibits a critical subset of the characteristics of a typical natural system. In this research, these characteristics are the presence of a fractal dimension combined with a "rich" spectral signature.

HYPOTHESIS 2: The combination of fractal dimension and spectral signature may be used to characterize the behavior of a pseudo-physical system.

Strictly speaking, a natural system is any system which exists in nature. The honey bees and their hive are both examples of natural systems. An artificial system is a man-made system. Examples of artificial systems include digital logic, computer systems and control systems (with an artificial sub-system as a control element).

The importance of characterizing a computing network as a PPS rests in the fact that natural systems are able to operate close to their functional capacity. There is evidence that this may be possible in computing networks. For example, automobile traffic in congested areas often continues to flow even when the network of roads is close to its capacity. Note that an automobile is a quasi-artificial system since it is an artificial system that has a natural component (the driver). Thus, if computing networks are perceived of and controlled as pseudo-physical systems, it also should be possible to use a portion of the unused, excess network capacity.

When the ideas of computing networks and pseudo-physical systems are combined, a new paradigm emerges. This paradigm

captures the behavior of computing networks for the purpose of improved communication and processing capability. What remains is to establish a firm foundation for this research. In the remainder of this chapter, the work of several researchers is outlined. Their areas of interest contribute substantially to the basis of this work.

1.4 System-Level Models

In the 1980's, several researchers (Haken, Huberman, Rucinski and Tewksbury) applied system-level models to computing networks to understand better their fundamental nature. In order to transcend the problem with congestion, innovative, unconventional but workable approaches to this problem have to be developed. The framework for this work was outlined in [Rucinski, Drexel, Dziurla, 1990]. It is an amalgam of concepts in [Tewksbury, 1987], [Huberman, 1988], and [Haken, 1981]. A new model for complex, distributed systems was proposed. This model is based on a structural homology between analog networks and a homogeneous network of north-south-east-west (NEWS) processors. The four compass directions imply a communication port exists in each of these directions. Homology is used here to mean that there is a functional equivalence between the analog and digital networks. See Chapter 5 for a complete definition.

[Tewksbury, 1987] focuses on the computational efficiency of a massively parallel computer system. The architecture of the system again is a two-dimensional mesh, composed of

processors with north-south-east-west (NEWS) communications links. Tewksbury is concerned with the space-time uncertainties associated with distributed applications running on an MPCS with greater than 2^{10} processors. Specifically, his concern is the delay uncertainty associated with distributed task execution and inter-processor synchronization. A two-dimensional mesh is an appropriate architecture for such studies. As the "space-time" properties of such systems are relatively simple, they allow a focus on communication between processors [Tewksbury, 1987]. In addition, it is relatively easy to visualize and analyze events in a two-dimensional mesh. Creation of two-dimensional graphics is straightforward and commercial visualization software packages typically have three dimensional plot capability. Lastly, it is relatively easy to make the mesh a more specific environment. For instance, additional communication ports may be added to change the shape of the mesh and increase network capacity [Shen, 1993].

Locally executed tasks, those that do not require their results to be sent to other processors, do not pose a communication problem. However, when several processors, perhaps a large portion of the mesh, cooperate on a task, communication of their results to a coordinating entity becomes important. For instance, it is not trivial to establish efficiently (throughout the distributed computer system) when a cooperative task is done. The information at and about an individual processor's contribution to the task

is known locally without any time lag. However, even adjacent processors learn this information with some small time delay. Processors at a significant distance suffer from substantial phase delay associated with information arriving from varying distances. Therefore, locally current information versus out-of-date global information is a problem.

One architectural approach to task coordination is to put all activities under central control [Tanenbaum, 1989]. In such systems, a specialized control center processor monitors system activity and edicts local changes to such things as routing tables. The control center is able to determine least-cost routing and effectively communicate this information to its nodes. Moreover, congestion avoidance is possible. This approach has been used both in small-scale systems (e.g., a mesh) [Nickolls, 1991] and large-scale systems (e.g., the AT&T network) [Schwartz, 1980]. Obviously, the larger the mesh, the more difficult it is to coordinate the activities of the processors. As systems expand and load increases, the concept of central control becomes less and less viable.

Alternatively, the local processors can be programmed to adapt to network conditions dynamically. For example, the processors in Tewksbury's mesh route packets based upon a monotonically decreasing distance to their destination while avoiding unsuitable routing directions. As load increases however, determining least-cost paths and locating network congestion becomes more difficult. Distant congestion yields

out-dated information. Local information may mask distant problems. Some mechanism is needed to capture network activity. Tewksbury's paper addresses this issue in a limited fashion, but he stops short of proposing how mesh activity may be measured. Tewksbury states: "The appropriate manner in which to define network activity and propagate that activity measure to neighboring sites has not been resolved yet and is a subject of continuing investigation" [Tewksbury, 1987]. In reply to a recent electronic mail query about the problem, Tewksbury indicated that, although he had as yet published no further research in this area, additional results will be forthcoming in mid-1994.

1.5 Computing System Ecologies

There is an extensive body of work that compares distributed computing systems to natural ecologies. For example, Miller and Drexler compare computational "ecologies." Computational ecologies are defined as spontaneous order in computer systems, based on various mechanisms. Spontaneous order is defined as: "... a patterned system which develops without deliberate design" [Miller, 1988]. Miller uses the growth of crystals as an example of spontaneous order in natural systems. It is not apparent how atomic forces result in the spontaneous formation of the crystalline structure, yet the structure emerges.

The authors contrast "prey-predator" and "producer-consumer" ecosystems. The prey-predator ecosystem is

obviously non-cooperative. Prey develop better defense mechanisms while predators develop better mechanisms to overcome these defenses. In the computational environment this could be likened to hackers and system administrators. Hackers diligently attempt to gain access to resources while administrators build better ways to keep them out [Stoll, 1990]. In the producer-consumer ecosystem, producers and consumers cooperate to mutual advantage. A thin veneer of skepticism exists as a deterrent to both the producer and consumer. However, this barrier to cooperation is easily overcome. A good example in the computational environment is the use of high-speed networks. Network administrators eagerly provide resources and introduce their clients to new services. The user community cooperates in learning the new technology while spreading the good news to other users. This positive feedback loop creates a need for more network resources.

Both of these situations result in a stable ecology as long as neither of the parties becomes overly aggressive. In addition, there is a certain self-similarity of behavior associated with different groups in the ecology.

In [Huberman, 1988], game theory [Axelrod, 1984], [Smith, 1982] is used to describe the behavior of a group of interacting agents. An agent is an intelligent entity that interacts with a computing network. Agents may be human or artificial (a computer). These agents contend for a set of resources based on imperfect and time-delayed knowledge. The

agents use a set of strategies and associated payoffs to contend for the resources. They change strategies according to what other agents are doing. This strategy-switching leads to crowding. The crowding results in massive switching of agents' strategies from one strategy to another. In other words, the system of agents oscillates, depending on the delay associated with information transfer.

Huberman draws an analogy between the behavior of the well-known quadratic map [Baker, 1990], [Drazin, 1992], [Tufillaro, 1992] and his model. He submits that his model is of the same type as the quadratic map: a finite difference equation, whose right-hand side has a parabolic maximum. While he shows data from the quadratic map, at that point there were no results from his model. However, Huberman's ideas of interacting agents and his mapping of their ecology into the quadratic map equation contributed substantially to the computing network paradigm introduced in Chapter 3. The quadratic map is described more fully in Chapter 2.

1.6 Synergetics

Haken presents another example of ordered behavior. In [Haken, 1981] the author defines synergetics as: "... the study of phenomena where patterns change qualitatively and dramatically on macroscopic scales. Such patterns may refer to space, time, or processes in space and time." For example, if the temperature beneath a fluid layer is increased, a temperature difference is established between the upper and

lower layers. As the temperature is increased, the fluid develops a hierarchy of instabilities (rolls) - leading to a group of different temporal and spatial patterns [Haken, 1981].

Systems such as this may be described by the following equation:

$$\dot{q} = N(q, \nabla, \alpha) + F \qquad (1.1)$$

Where: q is a set of variables that describes the system,

N is a non-linear function which has a non-linear dependence on its variables,

∇ represents spatial derivatives,

α is a set of control parameters, and

F represents "fluctuations."

Haken states that attempting a general solution to this equation is a "hopeless" task. However, there seems to be a "pre-established harmony" in the solutions that do exist. Initially, control parameters may be adjusted with a resultant behavioral change at the micro level. A topological deformation may occur but the macro level behavior of the system does not change. In other words, qualitative behavior remains more or less constant while quantitative behavior varies considerably. At a "critical point" in the range of a subset of α, there is a macro level change in the system. An example of this is the range of behaviors noticeable in a balloon as it is inflated. When the amount of air inside the balloon reaches a critical level, a dramatic spatial change occurs (i.e., it bursts). Up to that point, the structure is

deformed but remains fundamentally the same. An equation similar to (1.1) is developed in Chapter 3 to describe the behavior of computing networks.

Furthermore, Haken describes a "slaving principle" that relates to the subset of control parameters mentioned above. At the critical points of system operation, a small subset of α is all that is required to cause the system to undergo the dramatic change in its behavior. All other control parameters are slaved to this subset. Note that the behavioral regions, within which the dynamics of the system remain constant at a macro scale, are called "critical regions." Critical regions of behavior exist in the COMSIM simulator described in Chapter 4. These critical regions are used in Chapter 6 to analyze the behavior of the system.

Haken mentions that rather interesting things occur as systems move from one critical region to another. A well-known example of this is the group of changes that occur in the local velocity (i.e., the velocity at a point) of a fluid between two axial cylinders when the inner cylinder is rotated at increasing speed. The power spectrum of the local velocity clearly shows periodic behavior (two predominant frequencies), three states which exhibit more and more spectral components and lastly a state which exhibits a broad, bandwidth-limited spectral signature [Gollub, 1980].

Equation (1.1) includes a component, $\mathbf{F}$, which describes the "randomness" present in $\mathbf{q}$. Haken indicates that formalizing this parameter leads to analysis of synergetic

systems by methods used in theoretical physics. This is a promising formalism for analysis of computing networks as well. Moreover, the idea of isolating randomness in systems is used throughout the simulators developed in Chapter 4.

1.7 Chaos and Order in Systems

Huberman, Haken and Miller allude to the link between order in systems and "chaos." However, a universally accepted definition of chaos does not exist [Drazin, 1992]. No one has fully captured the true meaning of the phenomenon. Elements of the reality have been put forth as definitions. Here are a few of them:

1. Sensitivity to initial conditions [Poincare, 1913]
2. "Random" behavior exhibited by a deterministic system
3. An asymptotic motion that is not an equilibrium point, periodic, or quasiperiodic (formed from the sum of two incommensurate periods - see Chapter 6) [Tufillaro, 1992]
4. The existence of a fractal attractor (see Chapter 6).

Perhaps, a chaotic system may be defined as one which is unpredictably predictable. It is "impossible" to predict exactly the behavior of a chaotic system. However, given a stable environment (system operating conditions are not critically altered), the behavior of the system never leaves a bounded, n-dimensional space. The weather is a good example of this. If man does not destroy the ecosystem within which we live, the weather will remain predictably unpredictable.

Certainly, in natural systems there is an order within chaos [Berge, 1984]. Huberman and Miller have shown that computational ecologies may well be chaotic. For example, the quadratic map is virtually universally accepted as a deterministic system which exhibits chaotic behavior. The link between Huberman's ecology and the behavior of the quadratic map implies that his ordered ecology is chaotic. Moreover, it is likely that Miller's ecological systems have cycles that are chaotic in nature. However, this subject is not mentioned in [Miller, 1988].

1.8 Thesis and Scope of the Work

The remainder of this dissertation is arranged as follows. In Chapter 2, formal techniques of Systems Analysis examine and expand upon the numerous concepts and disciplines that are a part of this research. Chapter 2 serves as a micro-level guide to the dissertation. Chapter 3 puts forward a new approach to the analysis of computing networks. This abstract model of pseudo-physical systems acts as a macro-level guide to the dissertation. Chapter 4 describes the simulation model used to characterize the behavior of a two-dimensional mesh of processors. Chapter 5 describes the metric used to characterize the mesh of processors as a PPS. The metric is a link between the functional description of the mesh and its simulation model. The major contribution of this research, the "discovery" of chaos in computing networks, is described in Chapter 5. Chapter 6 enumerates the implications

and impact of this research. A new paradigm for the design and characterization of computing networks, based on system-level thinking is proposed in this chapter. The paradigm contains quantitative and qualitative aspects. The conclusions of this research, including that chaos in computing networks implies order (yet unused - see [Stallings, 1991]), are presented in Chapter 7.

CHAPTER II

SYSTEMS ANALYSIS

2.1 Introduction

This chapter combines formal Systems Analysis techniques with several interrelated but distinctive disciplines. These include: system theory, mathematical analysis and computer networks. The basic concepts of the multi-phase Systems Development Life Cycle (SDLC), of which Systems Analysis is an integral part, provide a step-by-step micro-level guide to the rest of the dissertation as well as a framework for the systematic study of the pseudo-physical behavior of computing networks. In addition, this work is basic research in the nature of communication. A model is proposed that leads to practical application: efficient design and management of computer networks. A formal construct is employed, based on Systems Analysis, to better forge a link between these two related, but separated, aspects of computer networks.

Various authors have put forth versions of SDLC. While the number of steps may vary, the content of the SDLC is essentially the same. Much of this work originated with Wetherbe [Wetherbe, 1984].

Whitten, Bentley and Barlow characterize SDLC as: "... a process by which systems analysts, software engineers, and programmers build systems. It is a project management tool, used to plan, execute and control systems development

projects" [Whitten, 1989]. The nine phases of the generic SDLC are:

Systems Analysis

1. Survey project scope and feasibility
2. Study current system
3. Requirements definition
4. Select a feasible solution

Systems Design

5. Design the new system
6. Acquire computer hardware and software

Systems Implementation

7. Construct the new system
8. Deliver the new system

Systems Support

9. Improve the system

The survey, study, definition and selection phases constitute the *Systems Analysis* aspect of SDLC. Phases five and six constitute the *Systems Design*. *Systems Implementation* is formed by the construction and delivery phases. Lastly, the improvement phase is known as *Systems Support* in most SDLC models.

It is important to note that the separation between phases may be blurred. In practical terms, phases often overlap. Moreover, there is feedback to the system definition from one phase to the next. This results in a constant refinement of the problem and its solution.

2.2 Global Approach - SDLC

The survey phase is also called the feasibility study. During the survey phase the scope of the project is defined. This includes perceived opportunities, problems, goals and possible solutions. In an engineering situation, the output of this phase is a *feasibility assessment* report. This presents findings, assessments, recommendations and a preliminary return-on-investment study. At this point, management decides whether to commit resources to the project or terminate it.

The next phase is the study phase. During this phase the current system, if one exists, is studied. In lieu of a pre-existing system, pre-existing solutions to similar problems are studied. The point of this study is to determine whether the perceived problems or benefits actually exist.

The study is done by reviewing documentation and by interviewing the users of the current system. It is crucial that the end-users of the system provide feedback. Unfortunately, it is my experience that system designers often neglect to include their users in this process. As a result, the new system is scorned and remains unused.

The output of the study phase is either an updated feasibility report or a *problem statement*. This report is signed-off by the user's management. It represents a mutual understanding of what the problem is with the current system. At this point, management elects to continue with the project or terminate it.

The next SDLC phase is the requirements definition phase. The purpose of this phase is to specify the function of the new system. Quite often, system models, called prototypes, are produced during this phase. The purpose of a prototype is to convey a sense of what the new system will be like to the end-users. In a computer processing environment, this prototype would include sample computer interfaces and file formats. Requirements are documented in a *requirements statement*.

The last SDLC element of systems analysis is the selection phase. This comprises a high-level, conceptual brain-storming session. All "reasonable" alternatives are put forth for consideration. During the selection phase "make versus buy" decisions are made. The solution which offers the best combination of technical, economic, and operational (i.e., political) components is the one selected for implementation. The output of this phase is a *systems proposal*.

Systems design is composed of two phases: new element design and component acquisition. In an engineering environment, elements of the systems proposal are given to system designers. These people treat the proposal as a set of high-level specifications. Low-level specifications are now produced to match the systems proposal. Deliverables are produced according to these new specifications.

The purpose of the acquisition phase of SDLC is to formalize the process of new hardware / software acquisition.

During this phase, a request-for-proposal (RFP) or request-for-quote (RFQ) is sent to vendors for the new system elements. The vendor responses are reviewed and a selection is made of who will provide the elements.

The construction and implementation phases of SDLC comprise systems implementation. In an engineering environment, any hardware or software not purchased is created in-house. The entire new system is well documented and installed. User training is done during the implementation phase. As before, management signs-off on the completion of each phase. This is to prevent lengthy, unwanted entanglements.

Lastly, the new system is maintained and improved. Any bugs are documented. Small improvements are handled without the full SDLC process. Eventually, this phase presents opportunities for new SDLC projects.

2.3 Global Analysis - Research Perspective

The following different areas are involved in this complex and interdisciplinary research:

1. system models,
2. mathematical interpretation of artificial systems,
3. system visualization,
4. chaos (meaning and measurement) and
5. applications (congestion avoidance, routing and flow control).

In this section mapping is provided between SDLC and the

research program to develop a comprehensive "physical" paradigm.

The overall objective of this research was to investigate the existence of "chaos" in artificial systems. As mentioned in Chapter 1, artificial systems are typically constrained to operate in predictable, predominantly linear regions. As the activity level of artificial systems increases, they become increasingly non-linear. It was suspected that this non-linearity led to "chaos."

In comparison, many natural systems require a strong chaotic component to be healthy. For example, [Goldberger, 1990] has shown that the electrical activity of a normal human heart appears to be fractal in nature. The spectrum of the electrocardiographic signal has a characteristic 1/f shape. [Schroeder, 1991] and others associate this type of spectrum with chaos. Moreover, Goldberger has shown that, immediately before death, the electrical activity of the heart is often relatively periodic, not chaotic, with a narrow frequency spectrum.

Thus, the useful operating region of artificial systems may be extended by controlling their periodic, chaotic and random components. A classification of operating regions is given in Chapter 3.

In order to establish that chaos exists in artificial systems, a suitable system, with a suitable problem had to be found. Tewksbury's mesh of processors seemed like a reasonable choice since the system elements are relatively

simple and the problem with congestion in such systems is well known. These systems exhibit a variety of behaviors that eventually lead to saturation. The regions associated with these behaviors are easily visualized using multi-dimensional visualization techniques. Chapter 5 deals with measuring and characterizing behaviors of the mesh system.

Note that even though the mesh architecture is used for study here, the research scope can easily be adapted to other topologies.

2.3.1 Research Feasibility

To determine research feasibility, a prototype simulator "Piotr" (Piotr means Peter in Polish) was created as the Engineering Systems Design individual development project. Piotr is a Pascal-language program which calculates the potential distribution of an electromagnetic field, using the iteration method [Hayt, 1967]. For the two-dimensional case, assume that the space is divided into squares with side *h*. The potential at the corner of one of these squares may be found from:

$$V_0 \doteq \frac{(V_1 + V_2 + V_3 + V_4)}{4} \qquad (2.1)$$

Where $V_1 \ldots V_4$ are the potentials of the four points neighboring V_0. Equation 2.1 is used iteratively to find the voltages at all of the points. V_0 becomes exact as *h* approaches 0.

In Piotr, each of the points in the distribution is represented by a NEWS processor. Processors determine their potential by synchronously recalculating their voltage based

on the average of the voltage of their four neighbors. A global monitor detects convergence and halts execution.

Obviously, such calculations may be done on real mesh-connected computer systems such as the Masspar MP-1 [Nickolls, 1991], the Intel Touchstone Delta [Zorpette, 1991] or the Cray CM-2. However, access to such machines is difficult and available programming tools are limited. In addition, models are more versatile, practical (in terms of access and control) and provide a convenient mechanism for generalization of concepts. The reasonable alternative, then, is to use simulators.

The simulators, which are derivatives of Piotr, are described in Chapter 4.

2.3.2 Study Phase

The study phase of the this project involved a careful survey of papers in the five disciplines mentioned in section 2.3. Examples of work in each area are given below. For maximum clarity, the reference section is organized in a similar fashion.

2.3.2.1 Applications and Routing. As mentioned in Chapter 1, Tewksbury's main concern is communication delay, and therefore routing, in the mesh network. Schwartz and Stern [Schwartz, 1980] have written a good beginning paper on routing techniques used in computer communication networks. This paper discusses least-cost routing in the ARPANET, TYMNET and TRANSPAC networks.

ARPANET (Advanced Research Projects Agency Network) began

with four nodes in 1969. A node is any intermediate point of intelligence. A node may be a switch, a computational element or a combination of the two. ARPANET's purpose was to study and demonstrate computer resource sharing [Roberts, 1970]. It has evolved into two networks: ARPANET and the Defense Data Network (DDN). Both networks use the same technology. ARPANET is research oriented. DDN is designed to meet Department of Defense goals for secure command and control [Stallings, 1991].

TYMNET was developed in 1970 by Tymshare, Inc. It was originally intended to provide cost-effective links from terminals to time-sharing computers [Tymes, 1971]. It has evolved into a general-purpose, terminal-to-host and host-to-host communications network.

TRANSPAC began service as the French public, packet-switched network in 1978. The network services of this network follow the X.25 protocol [Danet, 1976]. For a discussion of the X.25 protocol see [Stallings, 1991].

These networks use virtual circuit routing. A virtual circuit is a shared-space version of a circuit switched, fixed-space path from information source to sink. All packets in a message are routed along the same path. Circuit switched networks are similar to the telephone network. A dedicated communications path is established from end point to end point, usually through a set of intermediate nodes. In a circuit-switched network, bandwidth is guaranteed once the circuit has been established.

Ni and McKinley published a good review of various routing strategies in their recent work [Ni, 1993]. This is another excellent tutorial paper. The paper describes many basic packet-switched network concepts as well as input and output selection policies for nodes. Input selection is associated with selecting incoming packets for a place on the queue or one of the output ports. The output selection policy associates waiting packets with a particular output port. Essentially, this is the routing algorithm.

More germane to Tewksbury's problem is [Price, 1978]. Price presents simulation studies of networks operating in datagram mode. Tewksbury's mesh uses datagram routing. With datagram routing, the path each packet takes from source to sink is independent of others. This valuable paper introduces the principles of congestion avoidance:

1. congestion may be detected by monitoring the throughput (delivery delay) of the network and processor buffer occupancy (queue lengths),
2. in order to be effective, congestion avoidance measures must be applied at network data entry points, and
3. undeliverable packets must be deleted from the network as soon as they are deemed to be undeliverable (!). Note that buffer occupancy and delay are standard measures of network health.

A more recent paper, [Sushchenko, 1989], mathematically analyzes end-to-end delay in multi-linked virtual circuits.

An important result is: shortest delivery time is always achieved when packets for farthest-away destinations are routed first. To some extent this is a concept of the Lee routing algorithm which incorporates timing [Lee, 1992]. This inverse priority scheme makes sense. The algorithm avoids loading nodes with packets that take longest to reach their destinations. Local packets should arrive home quickly. However, starvation is a possibility for local packets.

Kim and Reed use the terms *temporal* and *spatial locality* to describe routing patterns in hypercube networks [Kim, 1988]. This is a formalization of concepts put forth by Tewksbury. Moreover, the authors introduce a hybrid routing technique, based on a combination of local and global information. Their dynamic algorithm routes packets to neighboring nodes that have shortest queue lengths. They claim that good performance is achieved when large weight is applied to local information. The COMSIM and DECSIM simulators, described in Chapter 4, use a version of this scheme. (Other papers, which deal more specifically with effective routing strategies, are covered in Chapter 6.)

2.3.2.2 Mathematical Interpretation of Artificial Systems. Chapter 1 provided an overview of Huberman's work in this area. Huberman derives mathematical relationships between competing agents and alludes to the fact that this type of quadratic equation is chaotic. However, Huberman does not demonstrate that his equations are "chaotic." That notion is an inference drawn from the well-known quadratic map.

The quadratic equation that Huberman refers to, and which does exhibit chaotic behavior, is:

$$x' = \lambda x(1 - x) \qquad (2.2)$$

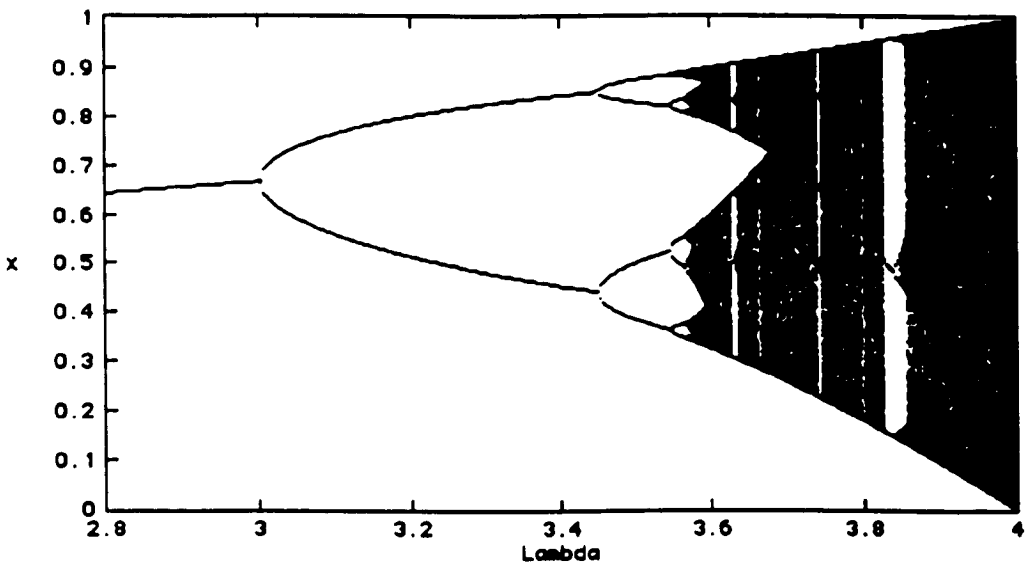


Figure 2.1 - Bifurcation diagram for equation (2.2).

Where: $0 \leq x \leq 1$ and λ of interest is $1 < \lambda \leq 4$. If we let $f(x) = x'$, and an initial value of $x = x_0$, then $x_1 = f(x_0)$, and $x_2 = f(x_1) = f(f(x_0))$, ... , $x_n = f(x_{n-1})$. x_1, x_2, ... x_n are iterates of this function. This sequence is called an orbit of x_0. If we let x_0 equal some arbitrary value, say .7, and iterate the function for n = 1024, the behaviors enumerated in Table 2.1, and shown in Figure 2.1, are exhibited by (2.2) as λ is varied.

Figure 2.1 is the bifurcation diagram for Equation 2.2. Bifurcation means "splitting." The limit values, for example x' after 1024 iterations, of a function or a system are used as the y axis coordinate. The x coordinate is the value of

Table 2.1 - Behaviors of the quadratic map.

λ	**behavior**
$1.0 \leq \lambda \leq 3.0$	period 1
$3.0 < \lambda < 3.449490$	period 2
$3.45 < \lambda < 3.54090$	period 4
$3.51 < \lambda < 3.564407$	period 8
$3.564407 < \lambda < 3.568759$	period 16
$3.57 < \lambda \leq 3.8285$	solid band
$3.83 \leq \lambda < 3.841$	period 3
$3.841 \leq \lambda < 3.848$	period 6
$3.848 \leq \lambda \leq 3.849$	period 9
$3.86 \leq \lambda \leq 3.999999$	solid band
4.00	chaos

some variable. In the case of the quadratic map, this is λ. Thus, the bifurcation diagram provides a type of graphic signature for the function of the system.

Period 1, referred to in Table 2.1 and indicated by the single value of x in Figure 2.1, when $\lambda \leq 2.966$, is known as a fixed point. Period 2 indicates that x' alternates between two limit values. In general, a system that exhibits periodic behavior endlessly repeats a finite set of values. If the system has period *n*, it outputs a unique sequence of values a_1 .. a_n. a_1 is repeated after *n* iterations. i.e., $a_{1+n} = a_1$, $a_{2+n} = a_2$, etc.

The solid bands in Figure 2.1 indicate that there are a very large number of different values for the function. Note that what happens in the solid bands, after the period

doubling, is not well understood. When $\lambda = 4.00$, the behavior exhibited by Equation 2.2 is known to be chaotic [Devaney, 1989].

[Martland, 1989] deals with the dynamic behavior of randomly connected networks of multiple input logic gates. These gates output a logic 1 depending upon probabilities associated with the input signals. For two-input gates, these probabilities are P_{00}, P_{01}, P_{10} and P_{11}. P_{00} is the probability that the gate outputs a 1 if both inputs are 0. P_{01} and P_{10} are the probabilities that the gate outputs a 1 if one input is a 1 and the other is a 0. P_{11} is the probability that the gate outputs a 1 if both inputs are a 1. The probability that the network outputs a one on the next time step is given by:

$$\rho' = P_{00}(1-\rho)^2 + P_{01}\rho(1-\rho) + P_{10}\rho(1-\rho) + P_{11}\rho^2 \qquad (2.3)$$

Where ρ represents the population of gates which had a one as an output on the previous cycle. With an appropriate choice of probabilities, this equation resembles the well-known quadratic map. Martland shows experimental results for various values of the probabilities. We made use of these ideas and measurement technique in the DECSIM model. The relevant discussion is given in Chapter 4. Message destinations in the probabilistic mesh model are patterned after Martland's work [Drexel, 1992].

Wheeler emulates Martland's work in a recurrent neural network setting [Wheeler, 1991]. The absolute sum of the weights within the modules is plotted against the system learning rate. Wheeler is able to generate bifurcation

diagrams quite similar to those for the quadratic map.

The important point here is that this simple system produces what appears to be "chaotic" behavior. Wheeler assumes that the bifurcation diagrams indicate chaos. Wheeler's diagrams certainly resemble those of the quadratic map. However, for verification purposes, a metric is needed which demonstrates, for data generated by this simulation, that the behavior shown is indeed chaotic.

Phillip Christie's work presents a good example of fractal behavior in an artificial system [Christie, 1993]. He shows that interconnections in silicon devices are characterized by a fractal dimension. "Fractal" implies shapes which exhibit self-similarity with irregular and fragmented patterns. Fractal objects have a dimension which lies somewhere between whole numbered values [Mandelbrot, 1977]. For example, a point has dimension 0 and a line has dimension 1. The Cantor set, which is an example of a fractal object, has a dimension between 0 and 1. This dimension may be measured in a number of ways. Chapter 5 deals with these topics in greater detail.

Christie indicates that a multi-fractal spectrum of interconnections exists in the hierarchy from the silicon to backplane levels. In other words, each two-dimensional layer exhibits a fractal connection structure. When the two-dimensional layers are combined, they result in a three-dimensional structure that is also fractal.

Christie's analysis shows that the system passes through

a version of Haken's critical regions. The notion of critical regions was introduced in Chapter 1.

Christie's work is important because it incorporates many ideas which are similar to ones introduced in this dissertation. Note how Christie's system is similar to, but different from, the mesh. While behaviors in the mesh are dynamic, i.e., they change over time, those portrayed in Christie's work are static in time and space.

Analogous to Christie's work is that done by Bedrosian and Jaggard [Bedrosian, 1987]. The authors show that the layout of large networks may have a fractal dimension. Essentially, they statically measure nodes (i.e., processors) per unit geometric area. This is similar to the correlation dimension technique which is provided in Chapter 5. In addition, this technique may prove useful in measuring packet deliveries per local node cluster.

Lastly, a paper by Musha and Higuchi [Musha, 1976] was useful for this research. The authors qualify and quantify the patterns inherent in traffic flow on a superhighway in Japan. They create a one-dimensional time series from the inter-arrival times of the vehicles. The authors draw a parallel with the continuous domain of the Navier-Stokes equation. Musha and Higuchi determined the power spectrum of their data. The spectrum has the characteristic bandwidth-limited shape of some chaotic systems [Gollub, 1980]. This work is important to the research in that it is an artificial system in which chaos has been found. The behavior of the

vehicles is similar in nature to that of datagrams in the mesh.

The measurements made of the traffic flow are one dimensional [Musha, 1976]. The mesh of processors may be considered to contain $N = 2n(n-1)$ of these one-dimensional systems. N is the number of input ports in a square two-dimensional mesh with n processors on a side.

Note that the system of Musha and Higushi is quasi-artificial. In other words, it is midway between a natural and an artificial system. Like packets, the vehicles are artificial entities. However, the routing is done by their driver, a natural entity.

2.3.2.3 System Models. Musha and Higuchi's work is similar in nature to the ground breaking research done by Shaw. He measured the time interval between the drips of a faucet and found chaos [Shaw, 1981]. This is a physical system (see definition in Chapter 1). The size of the droplets is continuous, within certain bounds. The time interval between droplets is a real number and therefore continuous. The droplet process is stochastic in nature. It can be approximated by a stochastic, continuous-time, continuous-space model. A complete discussion of system types and their models follows in section 2.3.3.

2.3.2.4 System Visualization. As mentioned in Chapter 1, Haken suggests that patterns of behavior of systems fit into measurable regions [Haken, 1981]. It should be possible to observe changes of behavior when certain parameters are

adjusted. [Christie, 1993] essentially makes the same statement. Some mechanism is needed, then, to present data from a model and determine where such regions lie. For this research it was decided to incorporate suitable graphical output capabilities into the simulators. In addition, IMSL/IDL™ and Mathcad™ software was obtained to display and analyze results.

2.3.2.5 Chaos: Meaning and Measurement. The Fast Fourier Transform (FFT), specifically the power spectrum, is often used as a measure of system behavior [Berge, 1984]. The FFT essentially is a signature of the system's behavior within one of Haken's regions. [Gollub, 1980] is often cited as an example of these signatures for a system going from periodic behavior to chaos. Two other oft-cited works in this area are [Crutchfield, 1980] and [Fenstermacher, 1979].

The basic problem is that the FFT does not always clearly identify the behavior of the system being measured. As indicated in Chapter 5, a combination of the correlation dimension measurement scheme and the FFT provides much more meaningful information.

2.3.2.6 Study Phase Output. The output of the study phase is an updated feasibility report.

Figure 2.2 shows that the research relevant to the dissertation comprises the first three phases of the total project. Chapters 2 and 3 comprise the analysis phase. Chapters 4 and 5 comprise the development phase. Characterization of the models is described in Chapter 5 and

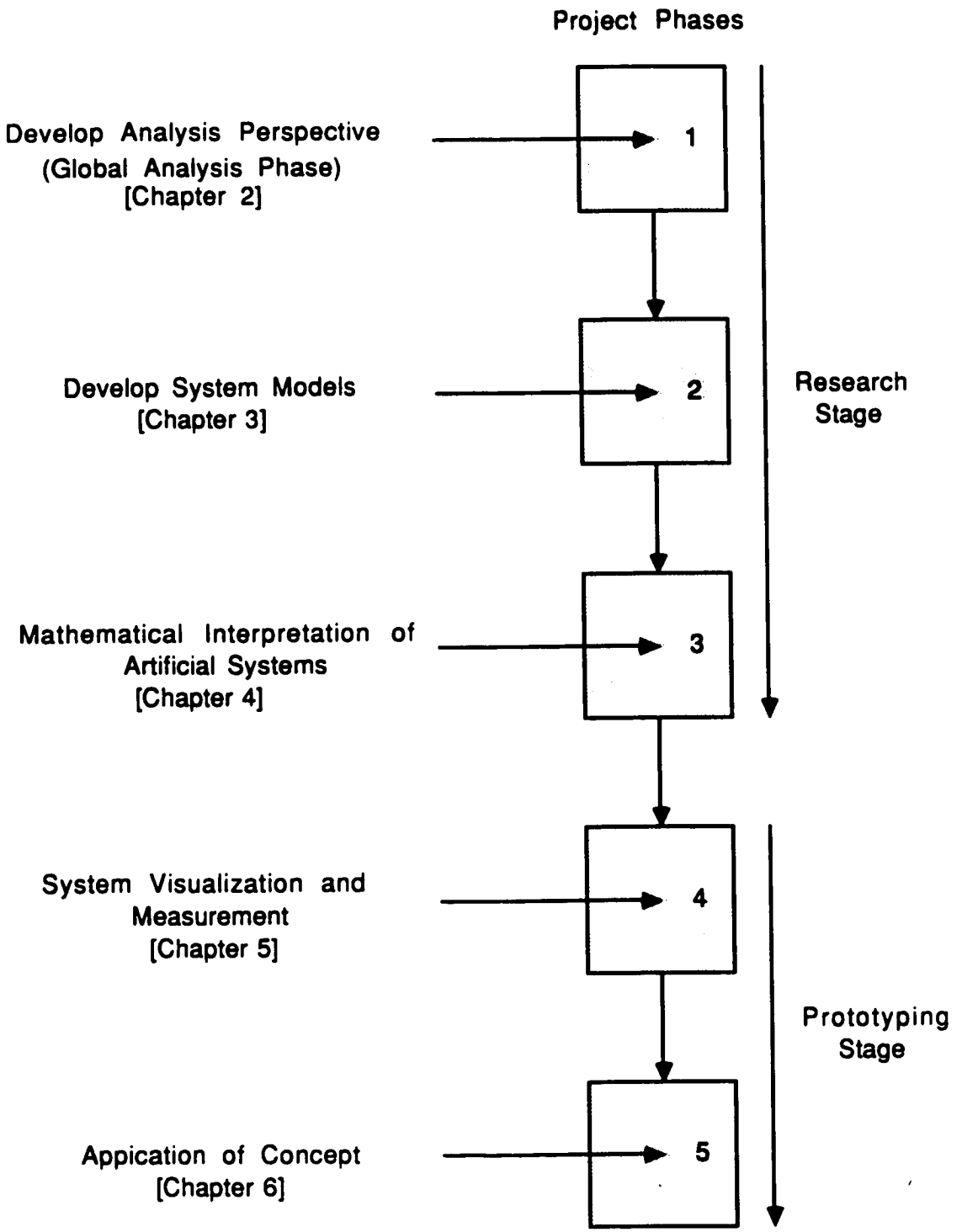


Figure 2.2 - Systems Analysis, Design and Implementation (adapted to the scope of this dissertation)

to a lesser extent in the development Chapters. Chapter 6 deals with the implications of the ongoing research.

2.3.3 Requirements Definition

A taxonomy was devised that characterizes models in common use by their salient features. It allows organized representation of the various modeling techniques and facilitates selection of the most appropriate to represent a two dimensional mesh of processors with datagram routing. The taxonomy of models is the output of the requirements definition phase.

2.3.3.1 Taxonomy of Models. The conceptual basis of models, defining homeomorphic or isomorphic relationships between the topologies of two systems, is discussed here because it is fundamental to many aspects of this research.

Models may be tangible (three-dimensional solid), mathematical (equations, computer programs), or graphical (two-dimensional) objects. Everyone is familiar with the use of models in such diverse disciplines as aeronautics, meteorology and architecture. Models are effective tools because they are abstract approximations of systems. These mechanisms are useful to capture and generalize the behaviors of systems.

Models may be used to confirm hypotheses about observed behaviors of systems which could be at a size scale very much larger (i.e., planetary systems) or very much smaller (i.e., molecular) than our own. At our own scale, models provide a convenient means for dealing experimentally with the real

system without going to the expense and difficulty of first building it. Moreover, models provide a means to predict long-term behavior (i.e., modified time scale).

Ideally, models should be based on all the "properties" of the systems they wish to approximate. Typically, however, they are not. The properties of a system are a combination of the parameters of the system and the relationships between the parameters. Only those properties of the real system which are the focus of the model, are most closely mapped to the model. In other words, those properties are held closely to their logical equivalent in the real system. In most cases, some limiting assumptions must be made in order for the model to be a tractable representation of the real system it approximates.

Mathematical models are of specific interest here. For a full treatment of this subject, see [Morrison, 1991]. In a few cases, mathematical models have exact analytical solutions. These are often simple, closed-form solutions to specific problems. What is more common is that the analytical solutions are extremely complex and thus require more than pencil and paper for their analysis. In such cases, computer simulation is used to implement the model. Thus, a simulation is the impression of a mathematical model onto a computer. Unfortunately, this process carries its own set of approximations and limitations.

It is useful to examine the factors that encompass different computer models. A taxonomy of these models (Figure

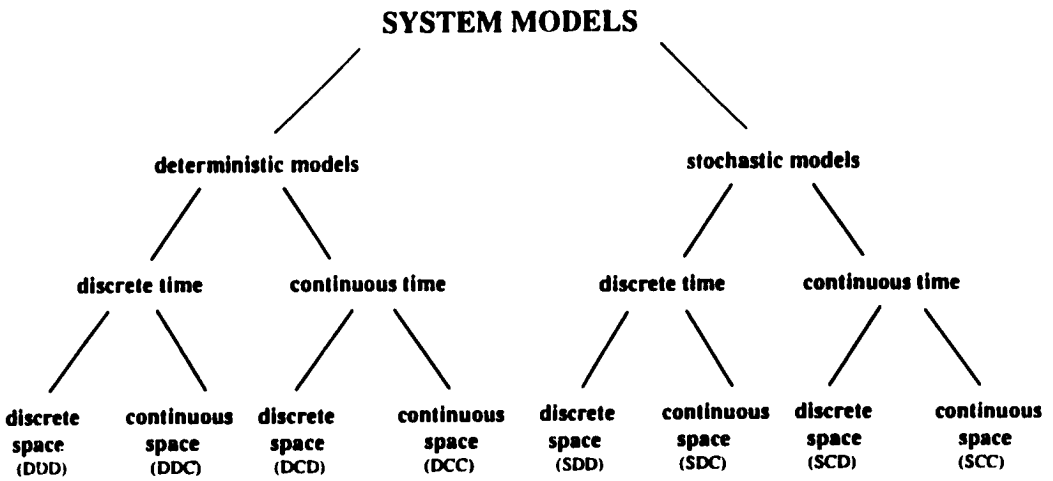


Figure 2.3 - Taxonomy of models.

2.3) may be formed by varying three fundamental properties: determinism, time-advance mechanism and system state. In terms of computer models, each of these properties is treated as a boolean variable. First, each of the properties is explored. Second, an example of each type of model is given. Additional model examples may be found in [Rucinski, 1990].

Models may be inherently deterministic or stochastic. A stochastic system is one which has at least one component that is random. In this case, random means there is a probability distribution associated with the element. A range of values and the likelihood of a specific occurrence is given. A coin flip is a good example of a stochastic process. A deterministic system is one whose output is uniquely determined once a set of input conditions is specified. Combinatorial logic is an example of this. It is important to note that most real systems contain both deterministic and

stochastic elements [Evans, 1988].

In a model, time may advance in a continuous fashion or in discrete steps. Continuous time advance is often associated with discrete event simulation. This approach allows for service time or idle time to be added to the current system clock value when an expected event occurs. Discrete time advance is inherent in many computer applications. Events are synchronized by the system clock. These models typically increment a time variable at the top of a system-function loop. All system activities take place in zero time during the time step. Distributed applications may be characterized by a combination of these two approaches.

Some systems, such as queues, have a finite set of discrete operating values. This state space is defined by the collection of variables necessary to describe the system at a particular time. Natural systems are most often characterized by large, possibly infinite, state spaces. The concept of system state is an outgrowth of the study of stochastic processes [Law, 1991]. Table 2.1 lists examples of the different types of models.

2.3.3.2 Deterministic Models. The COMSIM and DECSIM simulations used in this research are examples of deterministic, discrete time, discrete space (DDD) models. The system state is uniquely determined by queue length, queue occupancy, message destinations and the number of messages waiting to enter the system. The simulations may run in a completely deterministic manner. All operations are

Table 2.2 - Examples of models.

MODEL TYPE	EXAMPLES
DDD	COMSIM, DECSIM
DDC	LOGISTIC EQN, HENON MAP
DCD	MACHINE MODEL
DCC	ODE'S, LORENZ, PENDULUM
SDD	MARKOV CHAIN
SDC	PROBABILISTIC MESH MODEL
SCD	SLAM SIMULATION
SCC	REFLECTED PARTICLE

determined by a predetermined set of rules. However, both simulators use a coin-flip to route messages under certain circumstances. Therefore, these models are mostly DDD.

Good examples of deterministic, discrete time, continuous space (DDC) models are the quadratic and Henon maps. The Henon map is discussed in Chapter 5. It is easy to see that these systems are completely deterministic. Moreover, their state changes at discrete "time" increments and it can take any of a range of values.

Simple event-driven systems may be represented by deterministic, continuous time, discrete space (DCD) models. Imagine a bounded, single-server queue with deterministic arrival of two types of entities (D/D/1 in Kendall notation). The number of possible system states is fixed, based upon the maximum queue length (if the queue is full, entities are discarded). Assume that an entity arrives each time unit. Let the service time of entity type A = 3.14159 and that of

type B = 1.4142 time units. System time changes upon the completion of service for an entity. Thus, system time is a continuous variable.

Ordinary differential equations describe systems which are deterministic, have a continuous time variable and a continuous state space (DCC). The Lorenz equations (see [Gleick, 1978] and the damped pendulum [Baker, 1990] are well known examples of this type of system.

2.3.3.3 Stochastic Models. Computer networks are most often described by queuing theory models. These models are based upon Markov chains. The chains have a finite set of system states. State change occurs at discrete time intervals. It is based on the current state of the system and a set of stationary transition probabilities, i.e., the system possesses no memory. Thus, the models are stochastic, discrete time, discrete space in nature. Data Networks by Bertsekas and Gallager [Bertsekas, 1992] and Telecommunications Networks by Schwartz [Schwartz, 1988] are good introductory texts. These books introduce queuing theory, develop models for various network elements and estimate behavior based on the models. [Ahluwalia, 1992] is a recent example of the use of an SDD model. The authors present a discrete-time Markov chain model of the CM-2 connection machine.

The probabilistic model developed during this research [Drexel, 1992] adds a continuous state space to the above. For example, the continuous state space of the probabilistic

model is exhibited by the probability distribution associated with the number of messages present in a queue. Moreover, the transition matrices of the model are non-stationary. The probabilities are adjusted each time interval. This approach is used since it most closely approximates the system being modeled. The number of packets queued at a processor is allowed to affect the transition probabilities for that processor. Moreover, by using a transition matrix per processor, it is relatively easy to expand the size of the mesh.

An event-driven model of a manufacturing system is a good example of an stochastic, continuous time, discrete space (SCD) model. These systems are specified by probability distributions of arrival rates and service times. Typically, a finite number of entities may exist within the system at any given time. [Pritsker, 1986] provides numerous examples of such systems and their representation in the SLAM simulation language.

Consider a stream of particles with a Poisson distributed arrival rate. If these particles strike a rough surface, their angles of reflection are pseudo-random, with some probability distribution. This is an example of a stochastic, continuous time, continuous space (SCC) model.

The effectiveness of a model of a complex, distributed system improves when multiple leaves of the above taxonomy tree are included in the model. For example, systems typically represented by DDD models may well contain an

element of randomness. Adding this element to the model allows it to approximate the behavior of the real system more closely. When a single leaf of the taxonomy is chosen to represent a system, the overall effectiveness of the model may be impaired.

Moreover, in complex distributed systems, synchronization of events is difficult. This is especially true when the size of the system is such that direct interconnection of all system entities for accurate time measurement is not possible. For this reason, systems constructed at the VLSI level may not have this problem. However, the problem returns once the size scale of VLSI systems approaches that of interconnected computers.

Because homeomorphic representation, and ideally isomorphic representation, of systems is a fundamental requirement of the models used in this research, a new approach to their construction must be used. In this approach, individual sub-sections of the models faithfully follow the taxonomy, but the complete model is an amalgam of different model types.

Although COMSIM and the DECSIM router model are fundamentally DDD in nature, their routing algorithms are SDD to allow for greater flexibility. Thus, these models are a closer approximation of a real system. Due to the simulation technique used, the DECSIM model appears to be only a few steps removed from silicon.

2.4 Global Analysis - Systems Proposal

The last element of the Systems Analysis portion of the SDLC is the *selection phase*. At this point, a decision is made as to how the new system will be developed. For this research, this amounted to adopting a methodology for the remaining work. This subject is treated at length in Chapter 3.

CHAPTER III

COMPUTING NETWORKS WITH LOCAL ROUTING

This chapter puts forward a new approach to the analysis of computing networks. This abstract model of computing networks is consistent with both the Systems Analysis approach outlined in Chapter 2 and ideas in [Rucinski, 1990]. The chapter describes the form and function of computing networks at macro levels. In section 3.1, a two-dimensional mesh of processors is used to develop the concept of a computational energy surface. A functional description of the energy of the system is developed in section 3.2. Section 3.3 combines the concepts presented in the two previous sections into an abstract model of computing networks. This model acts as a macro-level guide to the remainder of the dissertation.

Sections 3.1 and 3.2 are derived from [Rucinski, 1990]. Figures are used with permission of the publisher.

3.1 Introduction

The initial discussion of the mesh found in Chapters 1 and 2 serves as a foundation for the next area of importance. That is, the impact of applications being executed by a distributed system. In applications, such as structural analysis or image processing, which produce thousands of concurrent pieces of data, it is not critical to obtain an exact (global) solution X, $X = \{x_1, x_2, \ldots, x_n\}$ where x_i is

the solution to be produced by the *i*th processor. Equally acceptable is an estimation $\Sigma \neq X$ such that $|\Sigma - X| < \varepsilon$ according to a certain norm $|\cdot|$ and error ε. Some components of Σ can be either computed incorrectly or simply missing. Similarly, timewise the velocity denoted by $\partial\{\Sigma(t)\}/\partial t$ of converging process $\{\Sigma(t)\}$ producing estimation Σ, can be slowed down as long as there are enough processing elements reaching consensus on the final estimation Σ before an assumed deadline T. In such cases, the process $\{\Sigma(t)\}$ for obtaining an estimated solution Σ is considered inherently dynamic, visualized as a pattern of individual estimates gradually emerging from a blurry computational surface. Moreover, processes $\{X(t)\}$ and $\{\Sigma(t)\}$ correspond to assumed and implemented algorithms, respectively. However, discrepancies in calculations are caused by other tasks being executed by the system, flawed processors, and scheduling problems.

It is assumed that the system is synchronous and guided by a distributed routing algorithm, inherently redundant in meshes. The driving force of the routing algorithm is specified later. The processors advance packets of messages and data files on every clock cycle. Each node may simultaneously issue ≤ 4 packets, one per compass direction.

3.1.1 Energy Fields

Processes in distributed systems, especially in two-dimensional "convex" structures, like meshes, can be visualized surprisingly easily if an analogy with potential theory is considered. The load of communication tasks to be

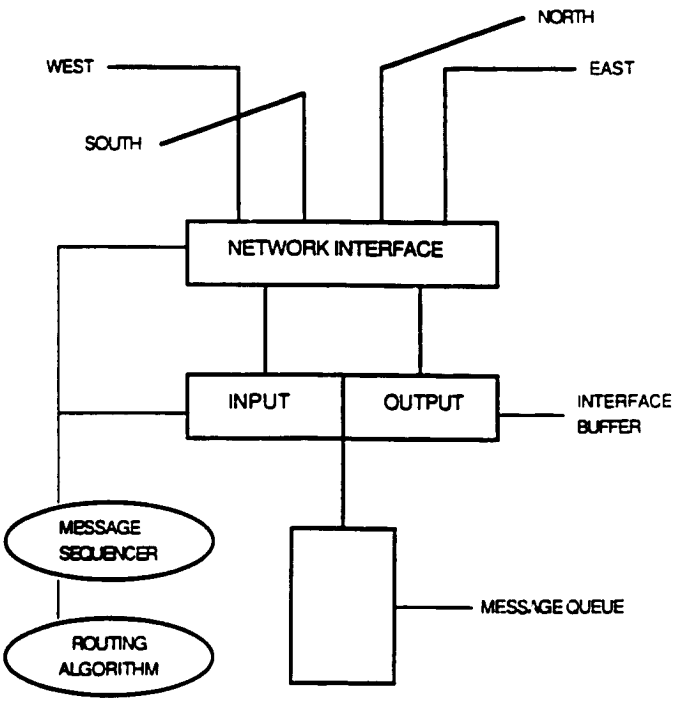


Figure 3.1 - Typical processor in a mesh network.

performed, associated with process {X(t)}, can be represented by *a field of potential energy* E(X,i,j,t) where i,j are the node coordinates on the grid. An example of a node is shown in Figure 3.1. Initially, at time 0, all task energy $E(X, i_0, j_0, 0)$ may be concentrated in a single node (i_0, j_0). The energy E(X,i,j,t) concentrated in any node (i,j) is a sum of potential energies contributed by individual packets

$$E(X,i,j,t) = \sum m_k \cdot d_k^p \qquad (3.1)$$

where:

(1) d_k is a destination address of a packet

(2) m_k is the cardinality of a packet set with the same destination address d_k

(3) p is the stress coefficient which determines the impact of destination addresses on the energy field

(4) k is the index for packets concentrated at node (i,j) with the same destination d_k at time t.

Since the available bandwidth is usually lower than the amount of energy (packets) to be distributed, there is more than one processor involved in process {X(t)}. The *image* I(X,{i,j},t) of process {X(t)} is defined as the convex area represented by nodes {i,j} participating in communication process {X(t)}. Consequently $I(X,i_0,j_0,0)$ is the point (i_0,j_0) itself. Once the process of scheduling begins, the image I(X,{i,j},t) expands with total value of energy E(X,{i,j},t) changed. The difference:

$$\Delta(X,\{i,j\},t+1) = E(X,\{i,j\},t+1) - E(X,\{i,j\},t) \quad (3.2)$$

is *a field of kinematic energy (power)* dissipated by the mesh during one clock cycle. This assumes no new processes have been generated at time t. The detailed dynamics of changes is discussed in section 3.2.

The speed by which energy is "consumed" by the mesh determines the communication activity level of the system. This parameter is, in fact, the power generated by the system. It is directly related to the *traffic* F(X,{i,j},t) in the system upper bounded by its bandwidth i.e., the number of communication links = 4n(n-1)), where n is the number of processors on the edge of a square mesh.

If p = 1 and no new processes have been generated at time t then $\Delta(X,\{i,j\},t) = F(X,\{i,j\},t)$. If E(X,{i,j},t) = 0 then process {X(t)} has converged. A local *deadlock* occurs if the energy of a processor is non-zero and traffic is zero. The

processor is deadlocked since no packets leave that processor. *Lifelock* occurs when the energy of a population of nodes remains non-zero for an extended period of time even though messages are being passed between processors in that population. In other words, if $E(X,\{i,j\},t) \neq 0$ and $F(X,\{i,j\},t) = 0$ local deadlock occurs. Lifelock is present if $F(X,\{i,j\},t) \neq 0$. $\{i,j\}$ is either the population of nodes, or a segment of image $I(X,\{i,j\},t)$. A processor may be engaged in several processes at the same time. Each process may be associated with tasks of distinctive nature and priority. For example, a processor can be involved in a computation, diagnostic, or control task.

3.1.2 Routing Strategies

Potential routing strategies naturally develop from the energy based model described above. Nodes collect some information about the status of neighboring nodes. They ignore the way the distributed and local data bases are generated. It is important to note that the desired local and distributed routing algorithms are reducible to linear models under light loads. But when entire clusters are congested, response times are determined by behavior in the non-linear regime. In general, the most attractive algorithm appears to be a distribution of relaxation "time constants" such that quasi-linear behavior is obtained on the small time scales during which actual individual routing decisions must be made. However, longer time constants are necessary to handle the development of significant regions of saturated flow. Also,

an increasing richness of alternative paths between end points is expected, particularly as the distance between end points increases.

Selection of a driving force for the routing algorithm in this model is based on information flow parameters such as velocity or information gradient; this is analogous to flows in physical systems. By choosing a locally dynamic routing scheme to proceed towards the energy field (activity and task load) gradient of the maximum descent, the impact of the packet to a local energy field is minimized. Thus, the probability of achieving the global minimal energy equilibrium increases. A forming network path is assumed to be positioned within a square whose two opposite vertices represent starting and destination communication sites. By avoiding hot spots locally, a routing path is likely to be "reasonable." The same physical model applies to both "activity-storage" and information fields. This research makes a contribution to the measurement of the information produced by differing types of activity in the mesh. Identification of laws between activity and information fields, analogous to the laws which exist between electric and magnetic fields, remains an interesting scientific challenge. See Chapter 5 for more details.

A packet is viewed as an elementary physical particle with a limited lifetime: initial buffering (conception), routing-started as a birth, and routing-completed as death. At each of these stages, a packet contributes in a different way to the energy balance of the network. A "healthy" network

can be characterized by a high rotation of short-lived packet population - fast relaxation followed by a frequent potential energy load increase.

Initially, the network is deterministic with a packet population assigned to each communication site. While the routing algorithm is being executed, the network manifests what appears to be "random" behavior because of an intensive interaction of messages which, in turn, increases delay uncertainties, a cause of non-determinism in an otherwise deterministic algorithm. Of special interest is non-deterministic behavior, which occurs in cases when there is a limited number of message interactions (small number of communication sites generating messages, small load tasks, deterministic and simple scheduling schemes, etc...). Not only because these cases are more easily handled, but because it is more likely that critical parameters which trigger network behavior from the predictable to the unpredictable can be more readily identified.

The main objective of the model is to deal with the extraction of information from network nodes that can be used in flow control and congestion avoidance. Since the routing algorithm is local, it is really based on the density of events (packets) related to a potential energy field E within a local region, whereas flow is based on the velocity of events related to a kinematic energy field *F* flowing through the nodes. Although the density may be easily measured (e.g., packets in region over some time interval), flow must be

estimated from densities and time intervals. Thus, the issue of "measuring" flow, given limited information available locally, remains one of the open problems in this approach. Moreover, a control problem exists with controlling global flow based upon outdated local information. The routing algorithm must:

1. be consistent with fluid flow and diffusion (relatively well understood physical analogues),
2. be consistent with good behavior under conditions of light loads, and
3. provide mechanisms for control as saturation is approached.

3.2 Dynamic Behavior

In this section a computing network with dynamic routing is defined and characterized as a "pseudo-physical system" (PPS). In general a pseudo-physical system is any artificial system which exhibits some of the behaviors of a natural system. In particular, a PPS is defined here as a complex artificial system which has a fractal dimension and a rich spectral signature. Pseudo-physical systems may be thought of as a form of iterated function system (IFS). The quadratic map, described in Chapter 2, is an example of an IFS. The concept of iterated function systems is developed further in Chapter 5.

A computing network, then, may be represented by a set of discrete-time equations. These equations may be either

deterministic or probabilistic in nature. These two representations lend themselves to two different views of the behavior of the computing network. This functional description of the network is developed in the next section. Further, the dimension of the network and its spectrum may be used to characterize its behavior. Linking the functional description to an overall view of the network as a PPS is addressed in section 3.3.

3.2.1 Energy-Balance Equation

The dynamics of the mesh are governed by the changing potential energy E(t), which corresponds to the activity load, defined as a field, $E(t) = \{E_k\}$, where $E_k = E_k(X_k, \{i,j\}_k, t)$ is the potential energy associated with communication task X_k with the corresponding population of sites $\{i,j\}_k$. In a deterministic case we have:

$$E(t+1) = A(E(t),C(E(t),E(t-1),\ldots,E(0),t),t)\cdot E(t) + B(E(t),D(E(t),E(t-1),\ldots,E(0),t),t)\cdot E^0(E(t),t) \quad (3.3)$$

Where: $A(E(t),C(E(t),E(t-1),\ldots,E(0),t),t)$ is the system matrix determined by the mesh connectivity with dynamic behavior control via *distributed routing algorithm* C, and $B(E(t),D(E(t),E(t-1),\ldots,E(0),t),t)$ is the system matrix determined by the mesh connectivity with dynamic behavior control via *distributed scheduling algorithm* D with boundary conditions: $\forall\ E(X_k,i,j,t) \le E_{MAX}$ (limited buffer capacity) and $\| F(t) \| \le 4n(n-1)$ (limited network bandwidth), where field F(t) is the total traffic in the mesh at time t.

All parameters in the equation above are non-linear

functions. Matrix *A* controls the relaxation process in the mesh when packets are progressing, energy is being dissipated, and old processes are disappearing. Matrix *B* determines the activation process when new processes with energy E^0 are being generated either externally or triggered by old processes.

In the probabilistic case the energy balance equation is transformed into:

$$e(t+1) = a(e(t),c(e(t),e(t-1),\ldots,e(0),t),t)\cdot e(t) + b(e(t),d(e(t),e(t-1),\ldots,e(0),t),t)\cdot e^0(e(t),t) \quad (3.4)$$

Where: e(t) denotes a normalized population of nodes active at time *t*, and *a* and *b* are stochastic matrices controlled by Markov processes *c* and *d* of order *t* respectively.

3.2.2 Simulation Model

Any attempts to describe a complex dynamic system analytically, even in cases where an approximation is appropriate, results in a closed-form formula so complicated (equations 3.3 and 3.4 for example) that it becomes useless. A computer program, used to simulate the system, is a more appropriate description. However, the energy balance equations form the functional description of the complex system.

An energy based model of the datagram mesh, driven by the distributed routing algorithm, is extremely difficult to describe analytically. Its behavior must be better understood before it is possible to analytically describe the mesh as a complex PPS, even with a reduced parameter size.

As described in Chapter 2, simulation is a promising

methodology. Such a simulation accepts a reasonable parameter space and is relatively easy to modify. The research is limited to synchronous networks, in which a simulation is a precise model of the actual algorithm. However, the datagram mesh can not be adequately modelled using only one leaf of the taxonomy described in section 2.3.3. A combination of deterministic and stochastic elements are needed for a realistic model.

Visualization techniques may be exploited in complex physical systems to monitor transitions from predictable behavior to complex system behavior. Visualization may be regarded as a combination of computer graphics and suitable interfaces that enable the user to transform complex, multi-dimensional data into visual representations. Visualization software is able to display data in three or four dimensions. Ideally, n-dimensional data is representable. For example see [Inselberg, 1990], [Grinstein, 1990] and [Taam, 1989]. In addition to the graphical output capabilities of the COMSIM simulator (see Chapter 4), IMSL/IDL™ and Mathcad™ software were used to visualize data generated in this research.

Several types of behaviors are anticipated in a complex PPS such as the mesh of processors. For example, Figure 3.2 shows an attractor space with regions of typical expected behaviors. The critical regions of the mesh are:

1. IDEAL EQUILIBRIUM - the mesh is fully loaded, in a steady state, behavior is linear, and may be described via the queuing theory; the surface of the mesh is

covered with a smooth flow of liquid.

2. SATURATION/IDLE - the mesh is totally frozen and resembles a solid, possibly a crystal.
3. LINEAR BEHAVIOR - the system is either linearly dissipative if relaxed (no new energies and old processes are being progressed) or linearly conservative if activated (more new processes are added). Energy changes in this domain are relatively small and slow.
4. FAULT-TOLERANT BEHAVIOR - the behavior becomes more violent due to reoccurring flawed nodes, but still linear; the system is able to adapt itself and functions at the degraded performances.
5. NON-LINEAR BEHAVIOR - this region is generally uncontrollable and governed by non-equilibrium

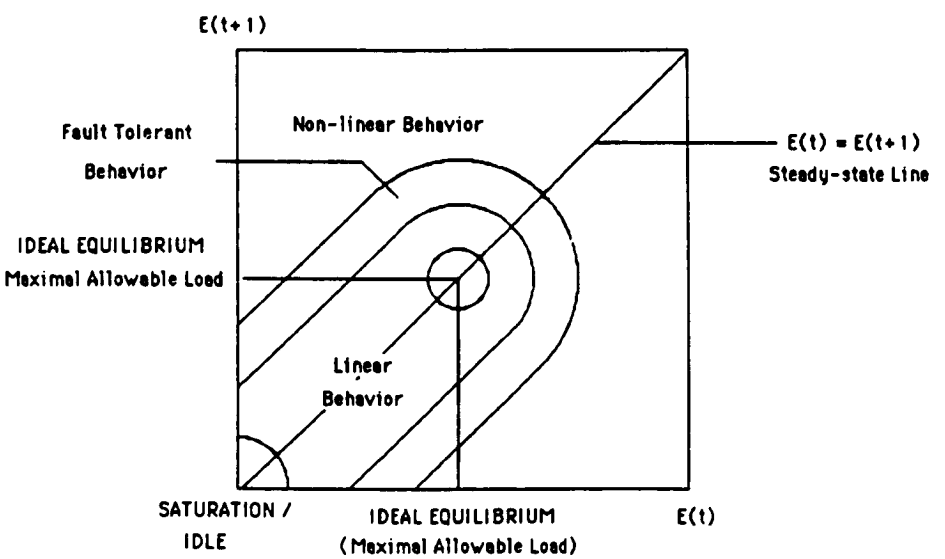


Figure 3.2 - The attractor space of the mesh with distributed routing.

phenomena; operating in this region is normally not permitted and believed to be catastrophic; analytical formal tools describing this area do not exist. Potential energy changes in this region are drastic, in both parameter and temporal spaces.

Visualization techniques were used to search for suitable mesh operating regions, in which transitions from equilibrium to non-linear behavior were likely. A three dimensional plot was used for this purpose. The data analyzed in Chapter 5 moves "up the slope" from saturation to ideal equilibrium.

3.3 Commutative Representation

The methodology discussed in sections 3.1 and 3.2 may be summarized as shown in Figure 3.3. The commutative representation consists of four nodes and demonstrable links between these nodes. The four nodes are:

1. **functional**, mathematical description of the system,
2. abstract, generalized, **probabilistic** model of the functional description,
3. concrete, tangible, function-oriented **simulation** of the system and
4. behavioral **signatures** which emulate the functional description of the system.

The important aspect of the commutative diagram is that it describes not only the mesh of processors but **any** pseudo-physical system. Whereas the functional description, probabilistic model and simulations may not be practical, the

behavioral model is. This model alone may be used to measure and predict the future behavior of a pseudo-physical system.

In general terms, links between the nodes are the indication of a "homeomorphic" or "isomorphic" relationship between the nodes. A homeomorphism preserves the underlying topology under an appropriate mapping. Thus, two nodes are homeomorphic if, given an appropriate topology, their behavior (excluding scale) is "similar". An isomorphism indicates that the relationship between the nodes is also reversible. Again, scale may be excluded. Results described in Chapter 5

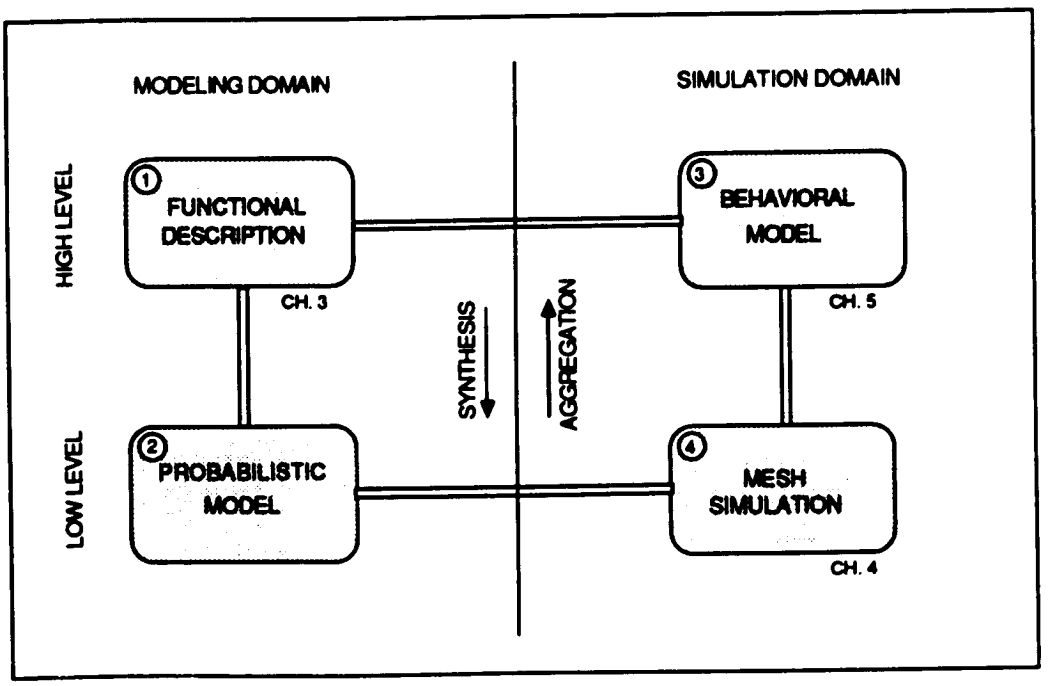


Figure 3.3 - Commutative Diagram.

demonstrate a homeomorphic relationship between the behavioral model and COMSIM.

As described in the taxonomy of models (section 2.3.3.1), a model is a mapping of one system onto another. In a sense, the model is an indication of the quality of the mapping. Moreover, homeomorphisms and isomorphisms qualify and quantify the relationship between two nodes. In some sense, the parameters and behaviors of interest in the model are extracted out of the larger space of the system being modeled. Another way of looking at this is that the behaviors of the model must be at least a homeomorphism of the real system.

If the mapping between the topologies of the real system and the model is adequate, the homeomorphism is bidirectional. That is, behaviors observed in the model should be observable in the real system and vice versa. With regard to the commutative representation of the methodology, behaviors observed in the node on one side of a link represent a mapping of those observed in the node on the other side of the link. Taking either the "high road" or "low road" yields the same qualitative view of the original system. Thus, the relationship between the nodes along a diagonal may be considered to be commutative.

3.3.1 Simple Example

Let us examine a simple example and see how this translates into the commutative representation. Damped pendulums with forcing functions are well known. The ordinary differential equation (ODE) representation of the pendulum

fits well into the upper left-hand node of Figure 3.3. The mapping connecting this node to the probabilistic model is the numerical method used to build a simulation of the ODE. Thus, the lower left-hand node becomes the ODE approximation realized on a computer. Within the computational limits of the computer (number of bits used to represent real numbers), an isomorphism exists between the functional description and the computer model.

The lower right-hand node of the diagram represents a detailed simulation of the mechanical workings of a real pendulum. The objective is to emulate an ideal device closely. Separate simulation elements would be necessary for the gears, weights, bearings, shaft, etc. The workings of this model may be observed and measurements taken as though the observer were looking at the actual device.

The probabilistic model is linked with the function-oriented simulation by static measures of their similarities and differences. These measures attempt to show how closely the generalization matches the concrete "reality." In other words, a set of parameters are held constant in both models while two or more are varied throughout their range. The resulting surface of similarities and differences can be analyzed with multi-dimensional visualization techniques. It is likely that a homeomorphism exists between the probabilistic model and the mechanical simulation. This is due to the difficulties described in the previous section. Note that the homeomorphism between the functional description

and the probabilistic model is also static in nature.

Dynamic measurement techniques are used to establish the isomorphism between the function-oriented simulation model and the behavioral model. This comparison is dynamic in the sense that time-series data are used to create signatures. In general, many measuring schemes may be used to qualify system behavior. Phase plots, time slices of messages per node and even plots of time series data have been used to observe what is happening. In this research, the unique combination of correlation dimension and FFT provides a consistent and accurate indication of the functional behavior of the simulation under various load conditions.

In this continuous system, the relationship between the simulation and the behavioral model is diffeomorphic. That is in the sense that the time-series data of the simulation maps exactly to the correlation dimension of the behavioral model and is reversible while maintaining the relationships between the derivatives of both systems. This idea is expanded in Chapter 5.

The same dynamic measurement techniques are used to relate the behavioral model and the functional description of the system. The intent here is to show that these two nodes have the same signatures under the same operating conditions.

For "simple" systems there is a known link between the functional description and the probabilistic model. The numerical methods used to model the pendulum's ODE's are an example of this. It is worth noting that even simple,

deterministic systems can exhibit complex behavior. For more complex systems, such as a mesh of processors, the relationship between these nodes is less obvious. As mentioned earlier in this chapter, a closed form, mathematical description of such systems is extremely difficult to achieve.

3.3.2 Link with Generalized View

The energy-balance equation in section 3.2.1 forms the *functional description* (Node 1 of Figure 3.3) of the mesh of processors. Ideally, the behaviors of the real system are completely captured in the matrices of the energy-balance iterated function system. Thus, there is an isomorphic relationship between the functional description and the actual system.

The probabilistic version of the energy-balance equation forms the basis of the *probabilistic model* (Node 2 of Figure 3.3) of the mesh. For analytic and computational tractability, the large set of Markov processes that represent the behavior of the entire system is replaced by a similar set of processes for each processor.

As mentioned earlier, the probabilistic model of the mesh [Drexel, 1992] is fundamentally an SDC model (stochastic, discrete time, continuous space). Arrival of messages, their acceptance or rejection by a node, the delivery of packets and node queue length are all probabilistic quantities. The transition matrix, used by each processor to determine changes in queue length, is non-stationary. That is, transition probabilities are recalculated during each time interval to

reflect changing conditions in the mesh.

Even though the probabilistic mesh model is fundamentally SDC, its routing algorithm is deterministic. This element of the probabilistic model more accurately reflects the behavior of the real system.

The COMSIM and DECSIM *mesh simulations* (Node 3 of Figure 3.3) are a form of the simulation model described in section 3.2.2. COMSIM and the DECSIM router are fundamentally DDD (deterministic, discrete-time, discrete-space) in nature. Packets are discrete data structures and are transferrable entities. Thus, a packet is delivered when it actually arrives at the input port of its destination processor. Processor queue lengths are a fixed, integer value. Packets actually occupy space in a processor's queue. In the case of the probabilistic model of the mesh, the number of packets in the queue is represented by a probability distribution. As in the probabilistic model, events in all processors are synchronized to the system clock. All processor activities occur once per time interval (clock tick).

Similar to the case of the probabilistic model, the COMSIM and DECSIM simulations are not entirely DDD in nature. The routing algorithm of these simulations contains a stochastic element. If a preferred routing direction is blocked, a "coin-flip" is used to choose an alternative. Thus, both are synergetic in that there is a stochastic element in their routing algorithm. These simulations are described in Chapter 4.

The *behavioral model* (Node 4 of Figure 3.3) of the mesh is formed by a dynamic (i.e., time series) mapping of behaviors of both the functional description and simulation model. Note that the mapping between nodes 1 and 2 and between 2 and 3 is static. In the case of the links to the behavioral model, the mapping is at least homeomorphic and possibly isomorphic in nature. The behavioral model is described in detail in Chapter 5.

CHAPTER IV

DDD MODELS - THE COMSIM AND DECSIM SIMULATORS

As described in section 2.3.3 (requirements definition phase) and specified in the Commutative Diagram (Figure 3.4), two deterministic, discrete-time, discrete-space (DDD) models were used in this research. The first, COMSIM (Communications Simulator), was an outgrowth of the work on Piotr. The second, the DECSIM router (Digital Equipment Corporation™ Simulation language), was created to confirm what had been done with COMSIM and to add data based on a set of "drive" conditions.

COMSIM is described first. This is followed, in section 4.2, by the discussion of the DECSIM router.

4.1 The COMSIM Mesh Simulator

Each of COMSIM's components is described in detail. This is followed by a description of one iteration (i.e., one clock cycle) of the mesh. The discussion of COMSIM is concluded by a section on the types of useful data extracted from the simulator.

4.1.1 Overview

COMSIM emulates the communication processes of a two dimensional mesh of identical NEWS processors. In the mesh, a population of processors, called originators, originate packets. These processors are preloaded with a batch of

packets before the simulation is started. These packets may have deterministic or random destinations. A deterministic destination means that the X and Y coordinates of the processor to which the packet is addressed does not change once the packet leaves the originator. Packets which have random destinations change their destination address each clock cycle. In some sense, these two types of packet addressing simulate: 1. the transmission of a file from one processor to another and 2. the transmission of a single packet message to a group of processors.

The mesh uses datagram routing. In other words, each packet is routed independently of any others headed for the same destination. Moreover, the dynamic nature of the routing algorithm continuously modifies the choice of "best path" for a given destination. Thus, as the load conditions of the mesh change, the paths used by packets to travel from originator to destination vary considerably.

COMSIM is designed to run until all packets have been delivered or the mesh "freezes." Freezing occurs when no packets move from one processor to another for three time units. In other words, there is no progress of packets toward their destination anywhere in the mesh.

Moreover, the simulator has the ability to "refresh" its originators periodically. In other words, a new batch of packets is put into the originator's queue after some time period T. This feature is useful for observing long-term behaviors.

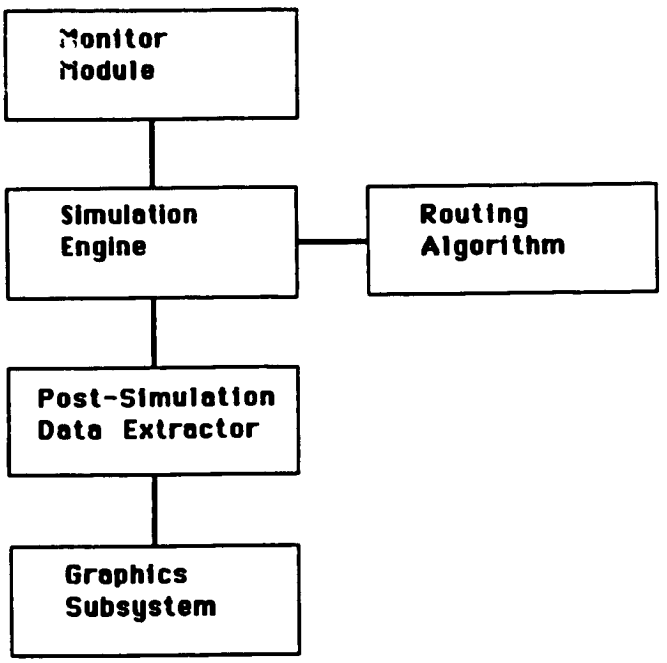


Figure 4.1 - COMSIM block diagram.

The simulator, shown in Figure 4.1, consists of several modules, each dedicated to a specific task. They are:

1. *monitor module* which specifies initial load distribution, ranges of operating parameters, data to be collected, and operating mode;
2. *simulation engine* which contains the data structures and logic of the processors, mesh, packets and iteration control;
3. *routing algorithm* which contains all of the

communications sub-tasks, including routing of packets;

4. *post-simulation data extractor* to collect spatial and temporal measurements and to provide a link between microscale (communication site) and macroscale (network) environments; and
5. *graphics subsystem* for displaying collected data in graphical form.

Certain COMSIM parameters are reconfigurable. For example, the number of processors in the mesh is established when the program is compiled. Moreover, the mesh may be square or rectangular in shape. A square mesh of twenty five processors was used throughout the bulk of this research.

4.1.2 Monitor Module

The monitor module is responsible for a diverse group of activities and parameters in the simulator. These can be divided into static and dynamic responsibilities. The static responsibilities include:

1. size of the mesh (X and Y limits)
2. processor queue length (number of packets)
3. initial mesh loading (which processors originate packets, number of packets, destinations)
4. message destinations (deterministic / random)
5. originator "refresh" (ON/OFF)
6. data collection (ON/OFF)
7. type of data to be collected (various types)
8. run time limit (number of clock cycles)

9. graphics display (ON/OFF)

These quantities are set with Pascal constant declarations. Thus, the variables are established at compile time and remain at that value until the program is re-compiled.

The dynamic aspect of the monitor is responsible for starting the simulator, collecting global data and housekeeping. Global data includes such things as an indication of how long it took for the mesh to freeze, average mesh loading and average packet delivery time.

The COMSIM engine is passed a set of variables when it is called by the monitor. These include the refresh rate (T) and "phase" angle (Θ). The phase angle is a timing offset used when more than one group of processors originates packets. In other words, processor group one originates packets at nT_1 clock ticks. Group two originates packets at ($\Theta + nT_2$) clock ticks, where n = 0, 1, 2, ...

The AUTOSIM feature of the simulator allows the user to collect data for pre-specified ranges of refresh rate and phase angle. For a set of originators with fixed batch sizes, iterates of refresh rate and phase angle often produce complex behavior.

4.1.3 Simulation Engine

The heart of the simulation engine is the Com_sim loop which is responsible for controlling all activities during one iteration of the mesh. These activities include:

1. clock control, including quit upon reaching time limit;

2. creating new batches of packets for originators if the "refresh" option is selected;
3. capturing processor-level and system-level data;
4. displaying temporal and spatial data;
5. calling the routing algorithm;
6. detecting a "frozen" mesh; and
7. interfacing with the monitor.

Each of these functions is a multi-level software package in itself.

The mesh consists of an array of Pascal records. Each element in the array is a processor data structure. It includes:

1. input ports for each compass direction,
2. output ports for each compass direction,
3. pointers to the top and bottom of the queue,
4. maximum queue length (constant),
5. packet acknowledgement (ACK) counter,
6. packet rejection (NACK) counter,
7. packet "generation" counter,
8. processor "voltage", and
9. neighbor's voltages.

Packets, which are dynamically created records, move from processor to processor. If accepted (ACK'ed) by a receiving processor, they are put in that processor's queue. Packets are rejected (NACK'ed) if the receiving processor's queue is full. In other words, the number of packets in the queue is equal to the maximum queue length for that processor. Packets

in the queue are forwarded to neighboring processors during subsequent clock cycles.

The processors use input ports to receive packets and output ports to send packets. The ACK and NACK counters are used to measure traffic in the mesh. A freeze occurs if no ACK's are counted anywhere in the mesh for the past three clock cycles. Processor voltage is the sum of potential energies of that node.

Processors determine their voltage from the number of packets in the queue and the output ports. The voltage indicates the processor's activity level. See section 3.1.1 for additional information. The potentials of the processor's neighbors are used by the routing algorithm (see section 4.1.4).

Packet records contain the following information:

1. destination address (X and Y coordinates),
2. originator's address,
3. time of origination,
4. number of packets in the batch,
5. status (waiting, ACK, NACK), and
6. packet generation.

The time of origination is used to determine delivery time. Average delay is the mean of the delivery times.

Packet "generation", batch quantity and originator address are used to simulate the exchange of messages between two processors. Upon receipt of the first of a new generation of messages, the destination processor enqueues a new batch of

messages for the originator. The number of packets in the new batch is equal to the batch length found in the packet that just arrived. Depending on the level of activity in the mesh, several generations of messages may be enroute concurrently.

Processors are serviced by row and left to right in a row. The simulator is capable of using a random servicing order. However, this feature was disabled so as not to introduce any "noise" into the system. The importance of removing noise sources is discussed in the next section. Note that, as far as the processors are concerned, activities occur in each processor "at the same time." This condition is necessary to insure that packet delivery delay is uniform in all directions.

4.1.4 Routing Algorithm

The routing algorithm consists of three parts: send old, send new and receive. During the "send old" phase of the algorithm, the status of packets which had been sent during the previous clock cycle is checked. If they were ACK'ed, they are removed from the queue of the sending processor. If they were NACK'ed, they are put back in the sender's queue. Much of this routing algorithm is also used in the DECSIM model (see section 4.3.3).

During the "send new" phase, the algorithm attempts to route any packets that remain on the queue. This process is explained in detail shortly.

During the "receive" phase of the algorithm, two things happen. First, if any packets in a processor's input ports

are for that processor, they are "delivered." In other words, the packet has reached its destination and it can be deleted from memory. An ACK is sent to the neighbor from which the packet came. Second, other packets are accepted by the processor if there is room in the processor's queue. If the packet is accepted, an ACK is sent to the neighbor who sent the packet. The packet is put at the tail of the queue. If the queue is full when the packet arrives, the packet is rejected and a NACK is sent to the neighboring processor.

Packets are routed from one processor to another based on the packet's destination address and the activity level of the processor's nearest neighbors. Given the coordinates of the current processor, (i_k, j_k), and the coordinates of the packet destination, (i_d, j_d), the destination quadrant is obtained from

$$C_x = i_k - i_d \tag{4.1}$$

and

$$C_y = j_k - j_d. \tag{4.2}$$

For example, a packet traveling from (1,1) to (5,5) yields negative values for C_x and C_y). Since packets are routed so as to decrease the distance to their destination monotonically, the packet may be routed either to the east or to the south. A choice of direction is made by the routing algorithm. It uses the neighboring processors' voltage to choose a "path of least resistance." The packet is placed by the algorithm in the processor output port corresponding to the neighbor with the lowest voltage. This process is repeated until there are no more packets to route or all

available ports are full.

If the neighbors' voltages are equal, one of the two alternative ports is chosen pseudo-randomly or the choice is deterministically forced to one of the two directions. The pseudo-random choice may lead to "chaos" at a local, processor level.

Shaw has shown that a process that seems to be completely deterministic can lead to chaos [Shaw, 1981]. For example, using the formula:

$$x' = \mathrm{Mod}_1\ 2x \tag{4.3}$$

values of x can be iteratively calculated. (This is an iterated function system, similar to equation 2.2.)

A "noise" source exists within equation 4.3. Assuming infinite precision for the binary representation of x, after a short time the numbers produced by the calculation become unpredictable. The random behavior exhibited by this deterministic system fits Parker and Chua's description of chaos [Parker, 1987].

The pseudo-random routing case of the simulator makes use of a "randomize" function to choose a seed value and then picks a random number based upon the seed. This process is done each time an arbitrary choice must be made for packet direction. Assuming that the mesh of processors is evenly loaded with packets, a situation that is likely to occur with T and Θ set such that the packets from two originators do not collide, then many arbitrary routing decisions are likely. Therefore, noise is introduced at each processor. Under these

conditions local chaotic behavior may be observed. On a global scale, the dynamics of the system should be inherently chaotic.

According to Shaw, intrinsic noise remains in a chaotic system after all other noise sources have been removed [Shaw, 1981]. What he means by "noise" is uncertainty, rather than randomness. A major objective of this research was to find and then understand this intrinsic noise. It is created in part by the collisions of packets as they are sent from originators to their destinations. Extraneous noise sources were removed from the system including the pseudo-random sequence of processor service, described in the previous section.

COMSIM and the DECSIM router are DDD models because they have no explicit, inherent noise sources. It would be impossible to remove all such noise from a stochastic model.

4.1.5 Post-Simulation Data Extractor

The purpose of the data extractor is to capture local and global information for display and for storage in a file for use by other software. Local data consists of the variables saved in the data structure of each processor. The data structure is described in the previous section. The local data is eventually used for Scientific Visualization experiments. The goal here is to understand the behaviors described in section 3.2.2 at the micro scale.

Global, system-wide data consists of the following parameters:

1. Originator energy,
2. Mesh energy,
3. Total system energy,
4. Originator traffic,
5. Total ACK's,
6. Total NACK's,
7. Total traffic,
8. Delivery delay: average and maximum, and
9. Mesh loading: average and maximum.

For energy calculations, the energy per packet is:

$$E_{packet} = (dx + dy)^P, \tag{4.4}$$

where:

$$dx = | C_x | \tag{4.5}$$

and

$$dy = | C_y | \tag{4.6}$$

and $p = 2$.

C_x and C_y are found from equation 4.1 and 4.2 respectively.

Originator energy is calculated using the number of packets that remain in the originator's queue. This calculation is facilitated by a list of originators, developed by the monitor and passed to the data extractor. Mesh energy is the total energy of all packets in the mesh, excluding the originators. Mesh energy measures the number packets allowed into the mesh and the relative distance to their destinations. Total energy is the combination of originator energies and mesh energy.

Originator traffic is the simple sum of all packets of a

given originator that appear in the output ports of all processors during a clock cycle. In other words, it is the number of an originator's packets that were sent out during a clock cycle. Total ACK's and NACK's provide a measure of "positive" and "negative" motion in the mesh. Total traffic is a combination of ACK's and NACK's. It is a measure of system activity during a clock cycle. Under dissipative conditions the number of ACK's is very close to the total traffic value. In other words, the number of NACK's approaches zero.

Packet delivery delay is calculated whenever a packet reaches its destination. The origination time is subtracted from the current time to obtain the delay. The time a packet spends waiting in queues is included in this value. The average delay value is the mean delivery delay of all packets.

Mesh loading is based on the number of links (i.e., output ports) occupied during a clock cycle. Obviously, in a finite two dimensional mesh some processors have more available ports than others. This is taken into consideration when mesh load is calculated. As above, the average load is the mean of all load values.

The global information provided by the data extractor can be displayed by the graphics subsystem. The display capabilities of the subsystem are described in the next section. Global data is saved in files for analysis by Fast Fourier Transform (FFT) and correlation dimension software.

The FFT is often used as a standard measure of system

dynamics (see section 2.3.2.5). Unfortunately, the FFT alone is not always a good indicator of system behavior. That is why the combination of FFT and correlation dimension is used to analyze COMSIM's behavior. This analysis is covered in depth in Chapter 6.

4.1.6 Graphics Subsystem

The purpose of the graphics subsystem is to allow the user to observe and understand what is happening in the mesh. The plots produced by the subsystem are of two basic types: temporal data and spatial data. The temporal interface is described first, followed by a description of the spatial output.

The first seven data types, described in the last section, may be plotted versus time (i.e., X(t) versus t). The monitor allows the user to select any combination of the variables. The display is normalized using E_{max}. This is a calculated value which represents the maximum possible energy that the mesh can attain given the number of processors, queue lengths and numbers of originators.

Moreover, the subsystem supports several forms of "phase" plot [Tufillaro, 1992]. X(t) may be plotted versus X(t+τ) to observe possible regions of attraction. The delay values τ=1 and τ=3 have been found to be particularly useful and are incorporated in COMSIM. Lastly, any two data types may be plotted against each other to create Lissajous figures [Stout, 1962]. Examples of the X(t) versus t and X(t) versus X(t+1) plot are found in the next section (4.2).

The spatial output of the graphics subsystem is a form of scientific visualization. The objective is to observe simultaneously several processor-level parameters for all of the processors in the mesh. Thereby, behavioral patterns that are not observable with the temporal outputs may be noted. The spatial "time slice" displays queue depth and packets in output ports for all processors. This is indicated by a pair of numbers for each processor. The top number indicates the queue depth. The bottom number indicates total packets at the processor (queue depth plus packets in output ports).

A sample of spatial output is shown in Figure 4.2. Color

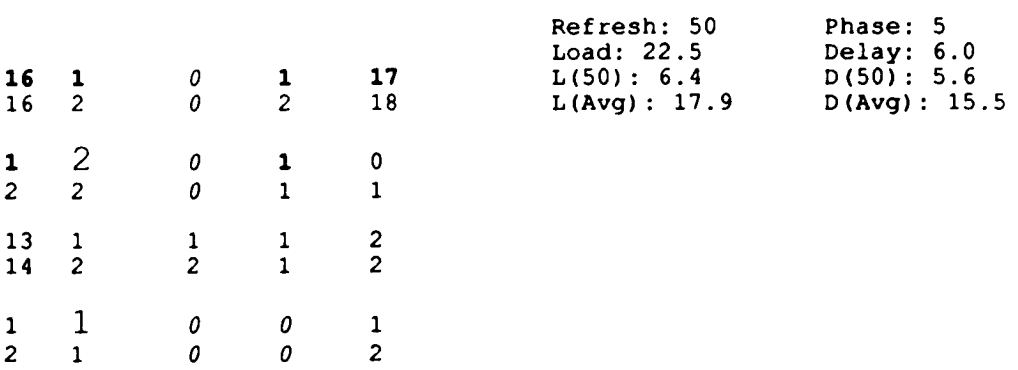


Figure 4.2 - COMSIM spatial display.

is used to indicate the health of each processor. Green, denoted by plain font in the figure, indicates that packets have been acknowledged recently. Red, denoted by oversize numbers in the figure, indicates blockage. Grey, denoted by italic numbers, indicates that the processor is idle. Light and dark shading, denoted by plain and bold font, is used to distinguish between packets from two different originators. For example, the upper, right-hand corner processor is one of

a group of four originators: the corner processors. This processor holds a total of 18 packets, one of which is in an output port. This processor's packets have been acknowledged recently.

Note that delay and load values are included in the display. The value 22.5 indicates that 22.5 percent of the mesh's ports were occupied during the last clock cycle. The delay value of 6.0 indicates that, on the average, 6 clock cycles were needed to deliver a packet to its destination.

4.1.7 COMSIM Implementation

Work on COMSIM was started in the winter of 1990, soon after the Piotr project ended. Most of the work was completed in one and a half years. It is important to note that COMSIM has undergone constant revision and enhancement since then. This effort occurred concurrently with the Systems Analysis, Methodology design, Probabilistic Model development, the search for appropriate measurement techniques and data acquisition.

The COMSIM simulator was written in Turbo Pascal; it runs on an IBM compatible personal computer. The Pascal language was chosen because of the availability of the programming environment at Plymouth State College (PSC), its ease of use and its portability from machine to machine.

At that time, computing resources available at PSC were limited. Therefore, alternative solutions were sought. The following model is one of them. The DECSIM Router is the other.

In the Spring of 1991, an account was obtained on the Cray Y-MP supercomputer, at the National Center for Supercomputing Applications (NCSA), at the University of Illinois. The reasons for obtaining this account were: 1. increased speed and 2. the possibility of running simulations of large meshes. Unfortunately, the COMSIM model could not be ported to the Cray because the Pascal language was not available. Therefore, a COMSIM-like model was created using the SLAM simulation language [Pritsker, 1986]. An advantage of this language over Pascal is its large number of available functions (i.e., queues, servers, etc.).

The model emulated COMSIM to some extent. Unfortunately however, the SLAM language is more suited to stochastic, continuous-time, discrete-space (SCD) models than deterministic, discrete-time, discrete-space (DDD) models. Moreover, the model was unwieldy. It was difficult to get packets to travel in two directions at once [Drexel, 1992]. SLAM wants its entities to travel left to right and top to bottom.

As a result of this "learning experience," it was resolved to obtain the best possible platform on which to run COMSIM (probably a high-speed PC).

4.2 The DECSIM Router

After several months of running COMSIM under different load conditions and observing its behavior by means of temporal data, X(t) versus X(t+1) attractors and FFT's, it

became apparent that independent confirmation of the data was needed. At that stage, the goals of the work were to:

1. independently confirm COMSIM's data,
2. collect additional data for a much enlarged mesh, and
3. search for different behaviors using a new set of mesh stimulus mechanisms.

The following is derived from collaborative work with Nicholas Ilyadis [Ilyadis, 1991]. It is included here with permission of the publisher.

4.2.1 Overview

Network routers, which utilize distributed routing algorithms, are used to implement the highest levels of a modern computer communication hierarchy. Routers implement the part of the communication channel which connects disjoint systems. The design of the routers, the routing algorithms and flow control mechanisms have a great impact on the throughput and efficiency of a network.

This section presents an investigation of a distributed routing network through the use of the DECSIM hardware simulator. Models of routers are used to simulate the distributed algorithm in a network governed by these routers. These models are based on the distributed algorithm used by Digital Equipment Corporation™ routers. This is important because DECSIM allows the writing of models that are concurrent in nature (as real hardware is).

Various load conditions are simulated on a mesh of 25 and 400 routers to characterize the network behavior both in the

linear and non-linear regions. Different scheduling methods are used to characterize the systems' bandwidth. Simulation results are presented and compared to the non-linear behavior of natural systems.

The simulations generated two measures of the network state: mesh energy and mesh traffic. In section 4.2.7 plots are presented that depict the various operating conditions graphically. The plots show total mesh energy versus cycles as well as total mesh traffic versus cycles.

4.2.2 DECSIM and its Use

DECSIM is a hardware development environment which provides languages that allow a designer to model, simulate, test and debug multi-level logic networks, ranging from simple gates to whole CPU's. The DECSIM languages include a structural inter-connect language, a behavioral modeling language and an interactive command language. The structural interconnect language allows the connection of "structural" components such as MOS gates, logic gates and flip-flops into networks that can be simulated. Larger models that have been previously compiled can also be interconnected, in a hierarchial fashion, to implement more complex systems.

Signals are the connections between structural and/or behavioral components. Signals can take on values of logic 1, logic 0, high impedance, and "undefined." States are variables that can be assigned an integer value within the bit range defined. The behavioral language allows the modeling of a block's operation in a software language rather than the

actual structural interconnect. This allows creation of quick turnaround models that have faster execution speeds than their equivalent structural counterparts.

The DECSIM behavioral language is structured like PASCAL and allows concurrent operation. The behavioral blocks communicate through ports. Ports are the points where external signals come into a software model. The internal program variables are states. The interactive command language allows the user to load a structural network that may contain structural and/or behavioral blocks and then simulate it. During simulation, the value of any signal or state may be examined, modified or saved to a log file. Command macros can be defined to encapsulate complex instruction sequences. The interactive command language can also be written in program flow style, similar to VAX DCL command files [Ilyadis, 1990]. The interactive environment runs in batch with indirect command files driving the simulation.

4.2.3 Router Model Discussion

The router model is written in DECSIM behavioral language. A block diagram is shown in Figure 4.3. The router has four I/O ports for data transmissions: 0 (right/east), 1 (up/north), 2 (left/west) and 3 (down/south). Each output port consists of a 48 bit Send Data (SD) output bus, a Request To Send Output (RTSO) signal and an ACKnowledge In (ACKI) input signal. The input port consists of a 48 bit Receive Data (RD) input bus, a Request To Send In (RTSI) input signal and an ACKnowledge Output (ACKO) output signal. The RTSO

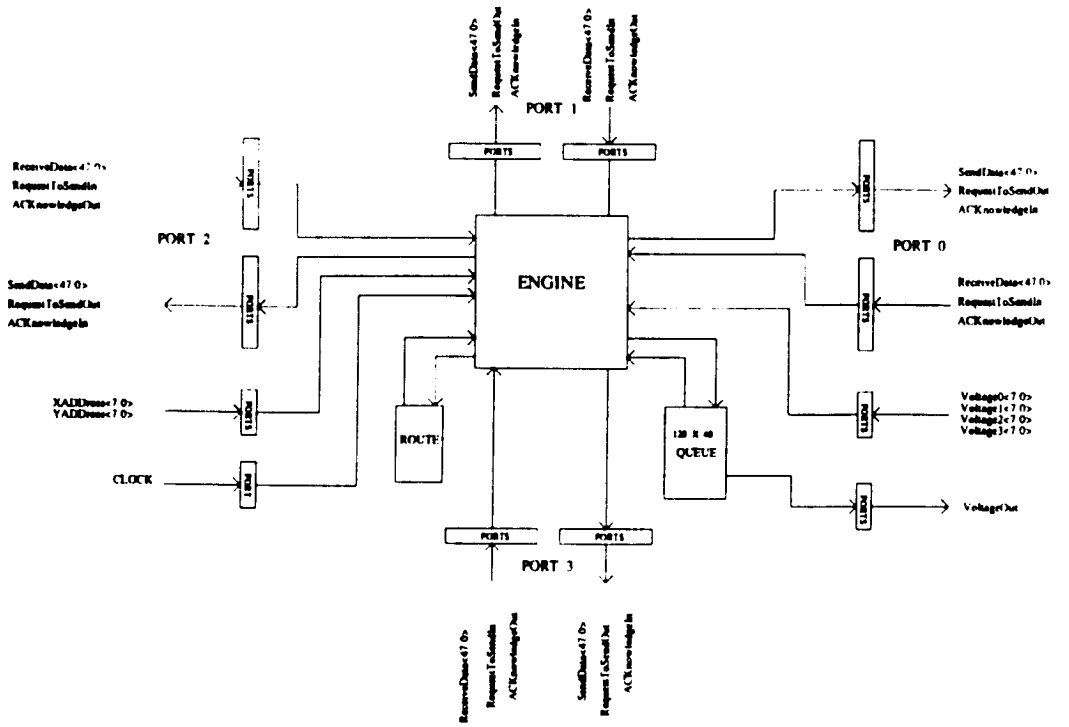


Figure 4.3 - DECSIM router model.

signal indicates that valid data is in the SD<48:0> lines. This is attached to the RTSI signal on the receiving processor. The ACKO signal is attached to the ACKI signal on the sending processor and indicates that the data was accepted. Each port also has a corresponding voltage sense input bus to receive the output potential of its neighbor in the indicated direction, i.e., $Voltage_0$ (V_0) for direction 0. There are a set of global signals that are port independent. XADDress and YADDress allow the X-Y address of the processor

to be programmed from the simulation. Voltage Out (VO) is used to indicate the processor's message load to its neighbors. The final input is the CLOCK. This is used to pace the processors and keep them synchronized. The program flow for the router is based around the routine ENGINE which is entered whenever the clock input changes. Sends occur during the high period of the clock, receives during the low period. The flowchart for the ENGINE routine is shown in Figure 4.4.

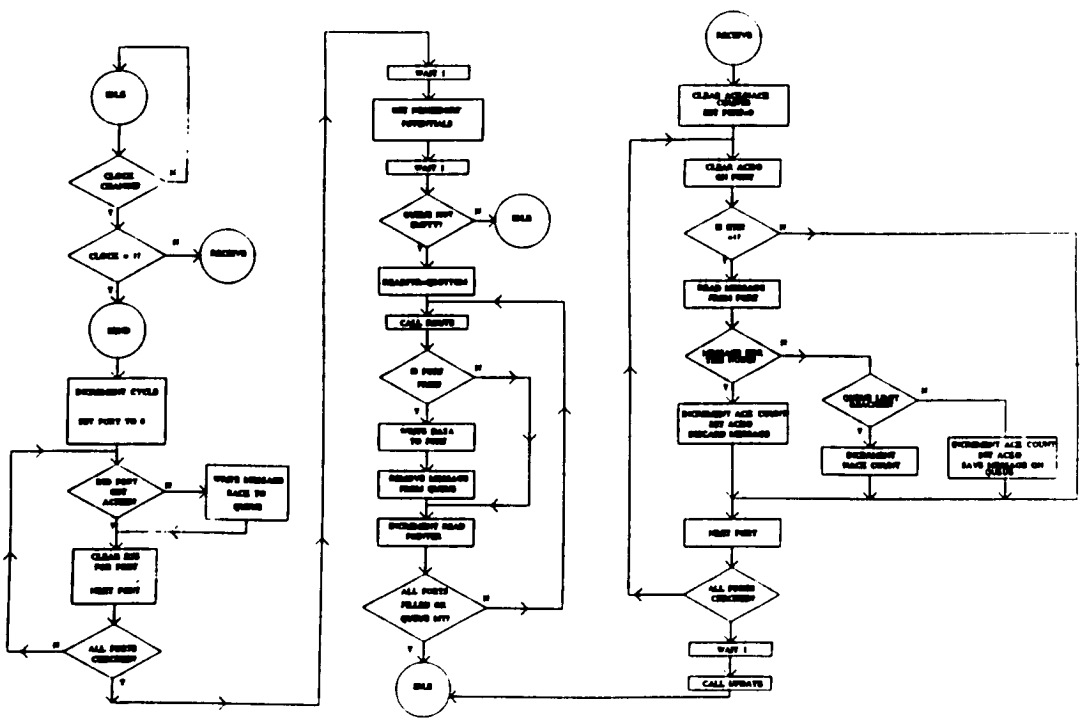


Figure 4.4 - ENGINE routine flowchart.

4.2.4 Mesh Development

The previously discussed router is the basic building

block for a mesh network. The work in [Rucinski, 1991] is based upon a 5 X 5 mesh of processors. The published results of this 25 processor mesh were used as a baseline to verify the operation of this router. The VALID schematic body of ROUTR was instantiated in a mesh as shown in Figure 4.5. This mesh is called TEST25.

All the processors have their address busses assigned unique names so that they can be driven to a unique value at the initialization of the simulation. The addressing scheme is in an X-Y coordinate system. As in COMSIM, the upper left hand corner has an X address of 1 and a Y address of 1. From this point on the notation is parenthesized as (X,Y). The X addresses increment as one progresses left to right, the Y addresses increment from top to bottom. The processor in the bottom right hand corner is designated (5,5). The boundary processors have their unconnected ports assigned signal names but not connected to any other device. These unconnected busses are driven with values at the initialization of the simulation. In the case of the RTSI signals a constant logic 0 maintains inactivity from that direction. The voltage sense busses are driven to their highest value, 255 in this case.

This schematic was processed by the G2S (GED to SPIDER) tool to extract a netlist. GED is a schematic editor. SPIDER is a netlist translator [Ilyadis, 1990]. This netlist, when processed through SPIDER, results in a DECSIM netlist that could be compiled into a simulation model. Once the model was available, simulations were performed that corresponded to the

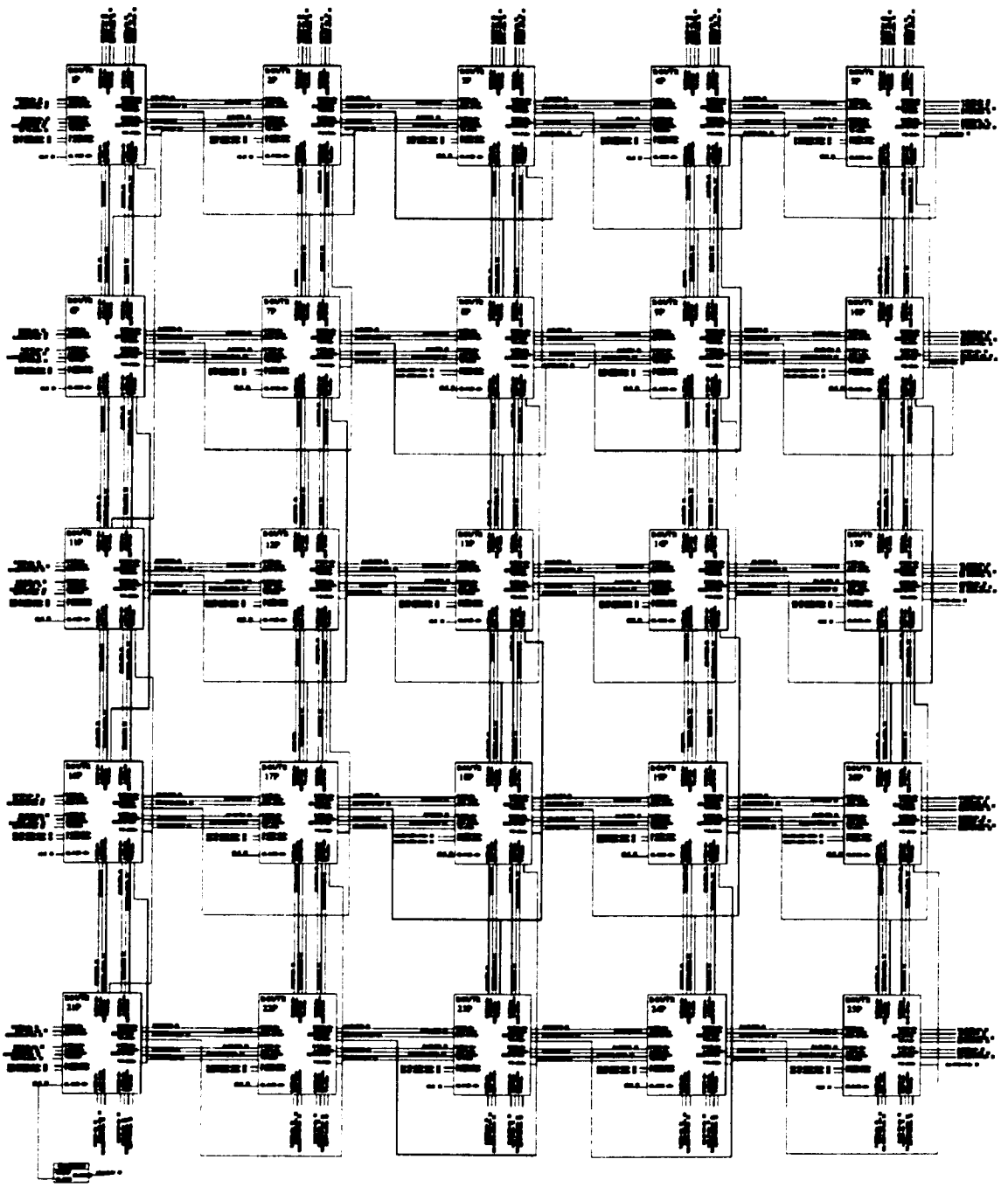


Figure 4.5 - 25 processor mesh.

routing experiments detailed in [Rucinski, 1991].

4.2.5 Computer Resources

The computer resources that were used on this project included a DEC VAXstation and a pair of VAX 8800's clustered together. The 8800 is a dual processor machine. The netlist processing (G2S/SPIDER) for the 400 processor mesh took approximately 1 hour on a single 8800 CPU with no other significant user load. The actual DECSIM compilation took 10 minutes. The resultant executable model is approximately 2.5 megabytes.

4.2.6 COMSIM Validation

This first simulation has processor (1,1) sending a group of messages to processor (5,5). In turn (5,5) is also sending a group of messages to (1,1).

This simulation required a command file that drove the simulation. DECSIM has an indirect command file feature that allows commands to be read in from an external file. These files can also contain command level state variables that can be used as global simulation variables.

Graphs of the mesh energies, traffic and energy attractors are shown in Figure 4.6. Figure 4.6a is a composite of several different energy measures. EMT is the total mesh energy, which includes the originating processors and their instantaneous message loads. EM is the total mesh energy of all processors less the two originators. E1 and E2 represent the energy of the (1,1) and the (5,5) processors respectively. The composite graph of the traffic measures is

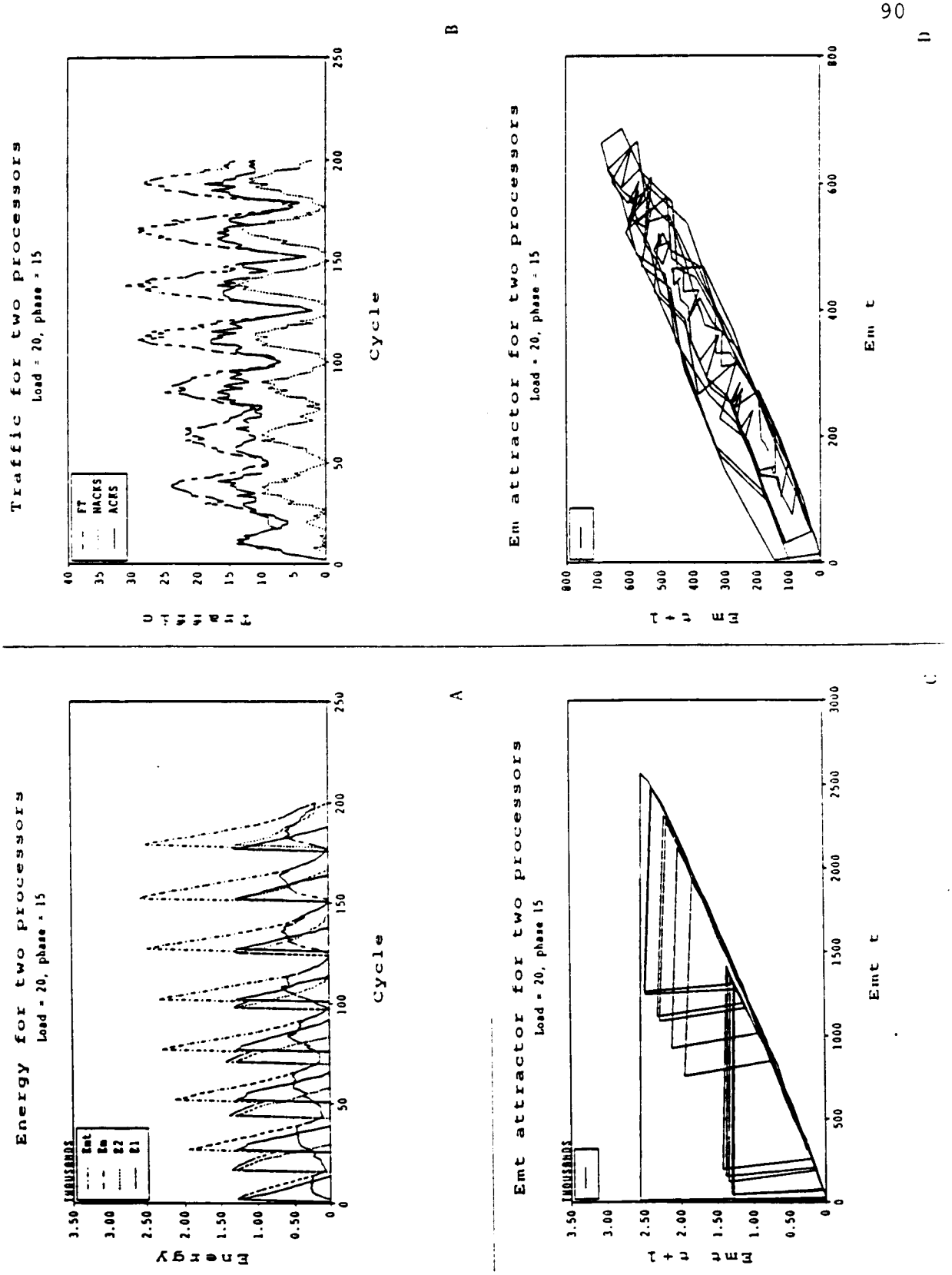


Figure 4.6 - COMSIM data validation.

shown in Graph 4.6b. This shows total traffic and its ACK and NACK components. Graph 4.6c shows the attractor for the EMT over 200 cycles. The discontinuities represent the introduction of message loads into the corner processors. The mesh is very dissipative with the majority of the operation occurring along the central E(t+1) = E(t) line. The EM attractor, in 4.6d, represents the mesh energy of the 23 other processors. These plots correlate very well with work in [Rucinski, 1991]. This exercise was meant to checkpoint the design so that it could be shown that a hardware simulation gave equivalent results to the software model. Some differences exist, due to the scheduling differences between the two simulations, but these do not affect the overall results.

An important aspect of the DECSIM version of the simulator is the environment's ability to create and handle larger mesh sizes. A 400 processor mesh was created using the 25 processor mesh as a starting point. The hierarchial structure of VALID and DECSIM allowed a 400 processor mesh to be generated without having to directly connect up 400 processors. The 25 processor mesh that had been generated up to this point was used as a hierarchial entity for the larger simulation. A model called FIVEBYFIVE was created from this 25 processor mesh. The connection points for the model were the boundaries of the 25 processor mesh.

4.2.7 DRIVE Series of Mesh Simulations

The unique contribution of the DECSIM router relates to

the way this mesh operates under load conditions that represent messages originating within the mesh. The task of loading the mesh with messages was challenging in that a scheduling scheme had to be designed that could show the dynamic behavior of the mesh under differing load conditions. After some trial and error, four different scheduling schemes were devised. These are designated DRIVE1 through DRIVE4. They range from most-deterministic to least-deterministic. Results from each of these experiments is summarized below. The accompanying figures include the following plots:

1. a temporal plot of Total Mesh Energy (EMT) versus Time (upper left),
2. a temporal plot of Total Traffic (FT) versus Time (upper right),
3. a phase plot of EMT(t+1) versus EMT(t) (center left),
4. a phase plot of ACK(t+1) versus ACK(t) (center right),
5. an FFT of EMT power spectrum (lower left), and
6. an FFT of ACK power spectrum (lower right).

As mentioned earlier, these plots are standard measures of system dynamics.

4.2.7.1 DRIVE1. DRIVE1 schedules 1/K processors per cycle to send a single message to their complement processor. The complement processor is defined as the processor mirrored across the center of the mesh. The complement can be mathematically derived as:

$$\text{destination X} = (\text{max\_x} + 1) - \text{My\_xadd} \qquad (4.7)$$

$$\text{destination Y} = (\text{max\_y} + 1) - \text{My\_yadd} \qquad (4.8)$$

Where: max_x and max_y are the maximum X and Y coordinates of the mesh. In the 25 processor case the complement of (1,1) is (5,5). In the case of an odd array size, the center processor sends to the (1,1) processor. The next cycle schedules the next 1/K processors. When K is not modulo of the mesh total there are some cycles when one less processor gets scheduled to make up the total. This continues until all processors have been scheduled. Once all processors have been scheduled the process begins once again.

Under DRIVE1 the 25 processor mesh freezes when K = 2. Free-running behavior is found for K ≥ 3. Figure 4.7 shows the 25 processor mesh with K = 4. The figure includes temporal, phase and FFT plots of total mesh energy (EMT), message acknowledgements (ACK's) and total traffic (FT). The temporal plot of mesh energy is relatively periodic. This is confirmed by the EMT FFT plot (DC component removed). The single spike at 32 represents the 1/K forcing function (32 = 128 samples / 4). The Y-axis scale of the linear FFT plot hides frequency components with lower power values. The phase plot (EMT(t+1) vs. EMT(t)) indicates what may be quasiperiodic behavior. The temporal of total traffic (FT), the ACK phase plot and power spectrum all show the evidence of system "noise." The amount of apparent randomness depends upon the size of the mesh, the value of K and the DRIVE scheme used [Ilyadis, 1990]. Note that the coin-flip feature is disabled for all of the plots in Figure 4.7. Similar behavior is exhibited in 400 processor simulations. Some smoothing is

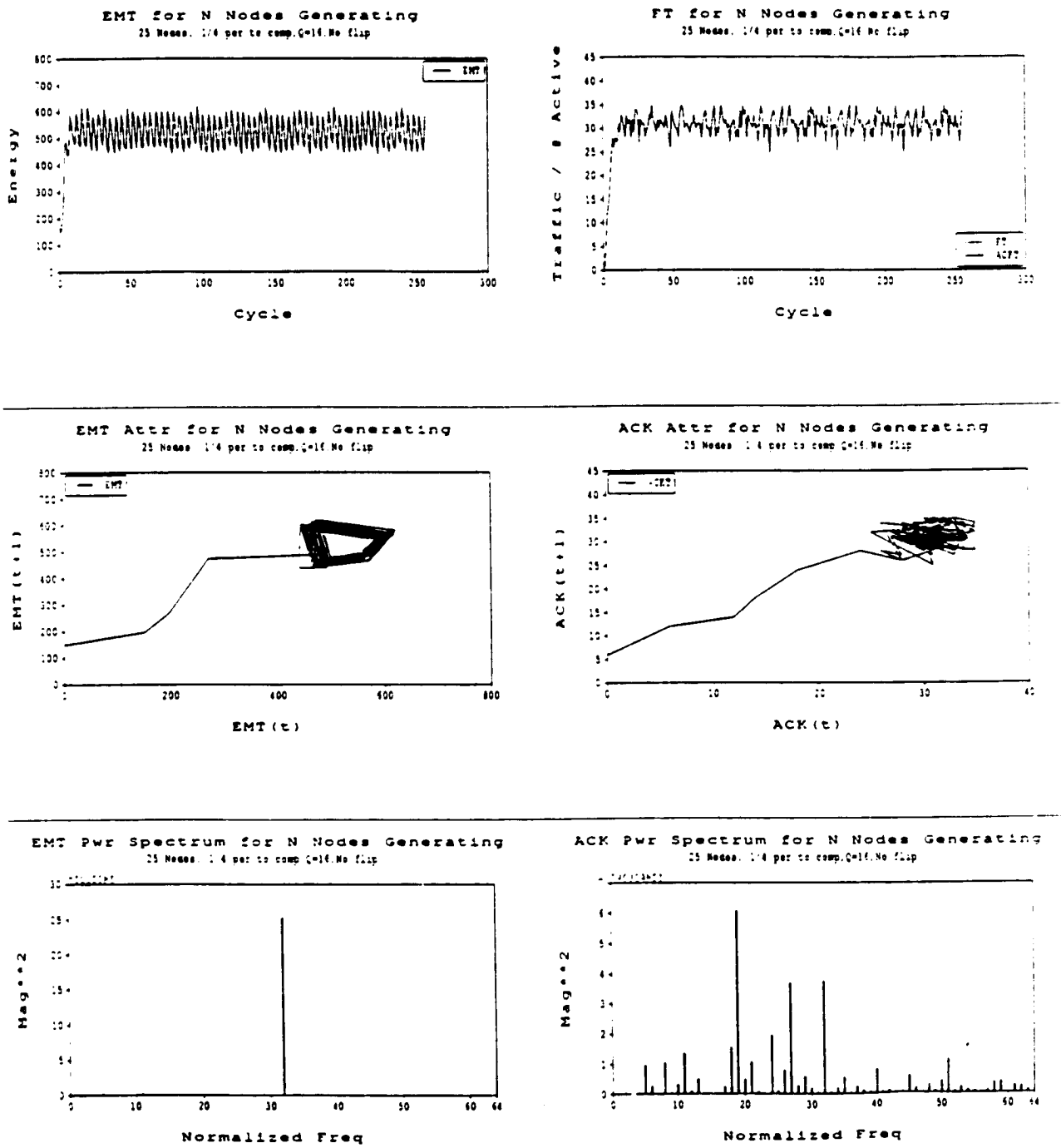


Figure 4.7 - DRIVE1 mesh simulation.

noted in phase plots and spectra (see [Ilyadis, 1990]).

4.2.7.2 DRIVE2. The DRIVE2 simulation maintains the same 1/K processor scheduling scheme but rather than sending to the complement processor it sends to a randomly determined processor. This randomly determined processor is selected at the onset of the simulation and maintained throughout the simulation. With DRIVE2 scheduling the 400 processor mesh freezes at K = 5. Free-running behavior occurs for K ≥ 6. The 25 processor mesh did not freeze for any value of K. Figure 4.8 shows the 400 processor mesh with K = 20. As expected for free-running behavior, the EMT temporal plot looks quite periodic. An increased number of frequency components is evident in the EMT and ACK power spectra. This is due to the random packet destinations. The two tall spikes in the EMT FFT plot again are due to the 1/K forcing function (6.4 = 128 / 20). The increased periodicity noted in the EMT plots is masked somewhat by the Y-axis scale used for the FT temporal plot and ACK phase plot. An increase in low frequency components is evident in the ACK spectrum. This is due to a higher mesh load level, made possible by the random packet destinations. An increase in low end spectral components with increased load is also noted in the COMSIM data presented in Chapter 5.

4.2.7.3 DRIVE3. DRIVE3 changes the scheduling scheme from the deterministic 1/K to a random set of processors. The number of processors that are scheduled each cycle is constant but the processors originating packets are determined pseudo-

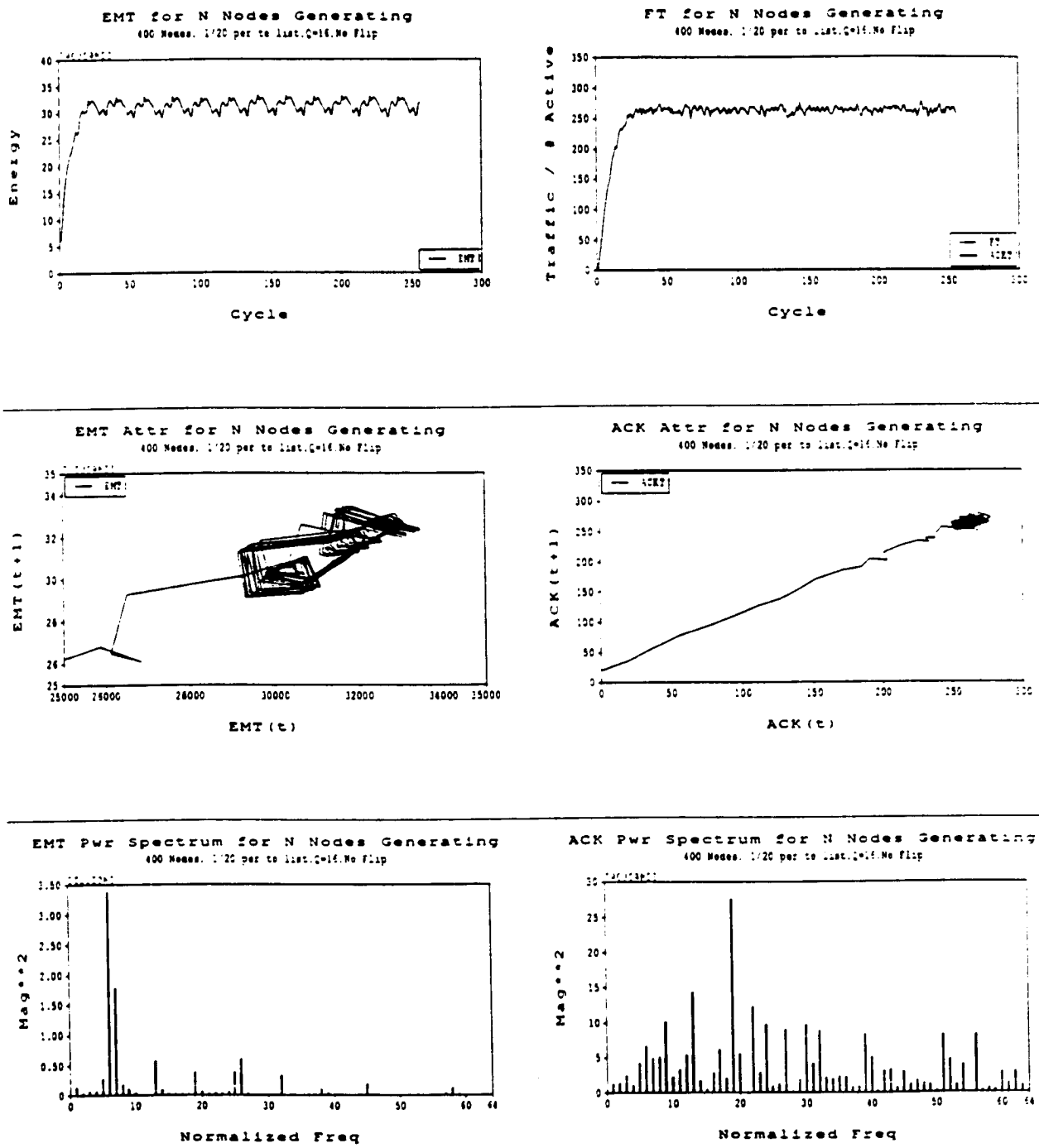


Figure 4.8 - DRIVE2 mesh simulation.

randomly. As in DRIVE1, the destination is the complement processor. Thus DRIVE3 has a random scheduling scheme that maintains a constant load of scheduled processors. The simulation results for DRIVE3 are similar to those of DRIVE1, for both the 25 and 400 processor meshes. The 400 processor mesh freezes at K = 12. The 25 processor mesh freezes at approximately K = 2. No appreciable change in mesh capacity is noted. Figure 4.9 shows the mesh moving from free-running operation to a frozen, deadlocked state. After 150 clock cycles EMT begins to grow rapidly. New packets are placed in originating processor's queues while no old packets are moving. The temporal plot of ACK's shows a steady decrease while FT remains constant. This is reasonable since ACK's are replaced by NACK's. The downward spiral of the ACK phase plot also shows the onset of deadlock.

4.2.7.4 DRIVE4. DRIVE4 is a combination of DRIVE2 and DRIVE3. It schedules a constant number of randomly determined processors that send to the same list used in DRIVE2. The results for DRIVE4 are similar to those for DRIVE2. The 400 processor mesh freezes at approximately K = 6. The 25 processor mesh does not freeze for any value of K. [Ilyadis, 1990] contains data for the DRIVE3 and DRIVE4 simulations.

4.2.8 Other Scheduling Schemes

Several other scheduling schemes were also simulated, but lack of time permitted neither analysis nor full simulation. These schemes included variations of DRIVE2 that determined a new random destination processor every cycle rather than using

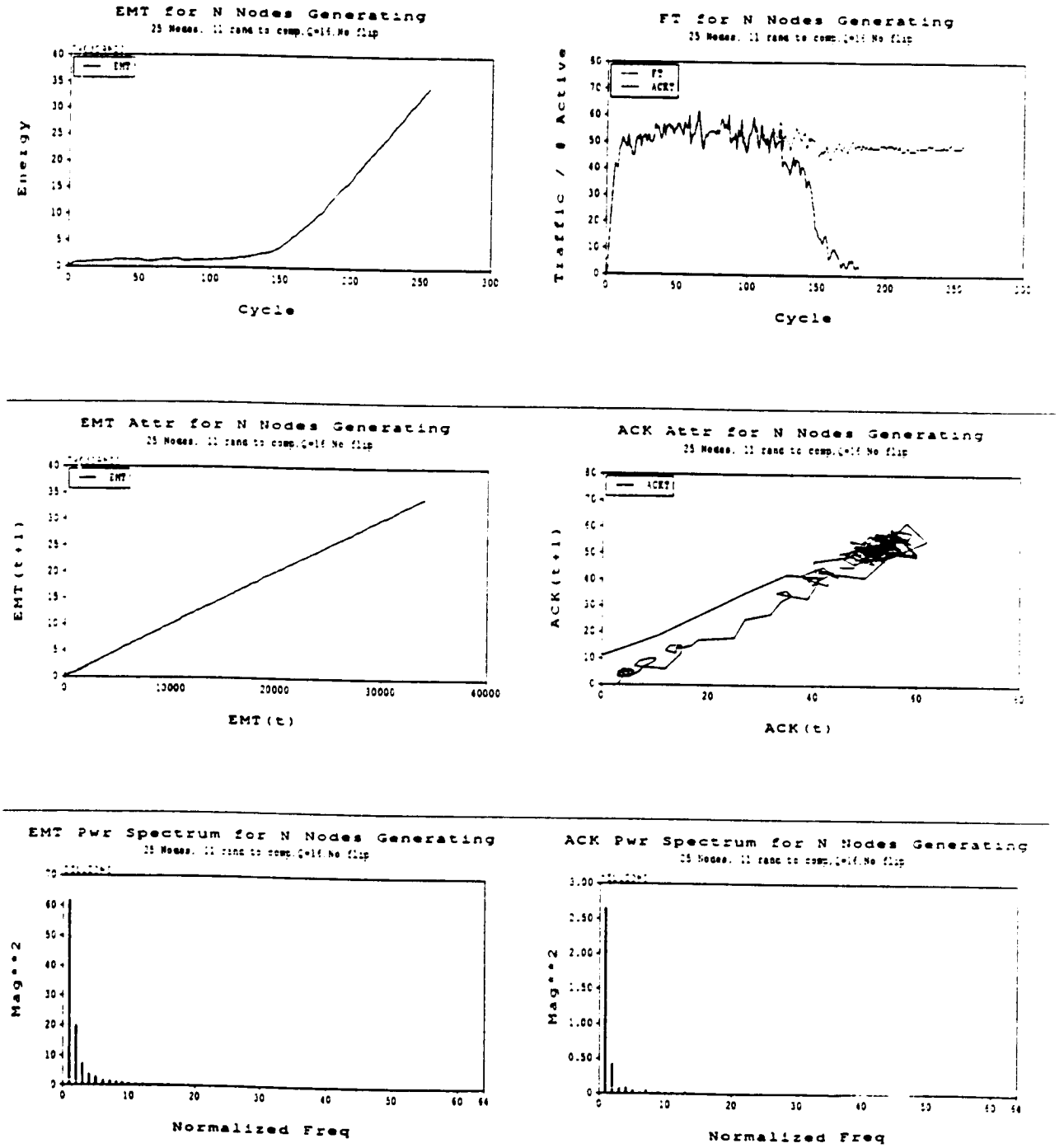


Figure 4.9 - DRIVE3 mesh simulation.

the list. Others included scheduling a cyclic number of random processors each cycle, or even a totally random number under some upper bound. Once the network was compiled it was very easy to modify the scheduling and destination schemes by using simple macros. This really shows the flexibility of the approach. But, the analysis and plotting of the tremendous amount of information generated by these simulations presented a problem. VAX DCL level command files could be written that ran DECSIM for each particular type of simulation under different load conditions. This scenario would generate a disk full of data that required reduction and analysis. Interactive runs could also be programmed in which network behavior could be monitored as the simulation ran.

The 25 processor mesh could be simulated for 256 cycles for a particular load condition in about 10 minutes. The 400 processor mesh unfortunately was much slower. The average run of a mesh that did not saturate was 8 hours. Simulations that caused mesh saturation ranged from 9 to 10 hours on a single CPU. Thus, the 25 processor mesh was useful in developing the scheduling schemes which could then be applied to the larger mesh.

4.2.9 Observations

As a result of the substantial amount of data collected, it was firmly established that the standard measures of system dynamics shown in this chapter are not sufficient to prove whether chaos exists within the mesh, nor are they sufficient to provide behavioral signatures for critical operating

regions. Tantalizing hints of interesting behavior lurk in the data, but the standard system metrics are insufficient to demonstrate it.

CHAPTER V

CORRELATION DIMENSION AND CHAOTIC NETWORK DYNAMICS

The contents of Chapter 5 are the result of collaborative work, to be presented at SIMTEC93 [Blaine and Drexel, 1993].

5.1 Introduction

In this section some techniques for quantifying the notion of "chaos" in dynamical systems are examined. The main tool used is the correlation dimension (CD). The CD has been in use for about a decade and has proven to be an extremely valuable tool for the analysis of complex experimental data [Grassberger, 1983], [Tufillaro, 1992]. It has not yet, however, become a standard technique of engineering practice, and the first portion of this chapter is devoted to an exposition of its properties. The results from CD calculations are compared with those from Fourier analysis and the implications are discussed.

The systems considered are discrete, i.e., they give rise to sequences x_1, x_2, ... which may be regarded as time series. These time series may consist of either real numbers or of vectors. Of particular interest are the iterated function systems (IFS), which are generated by a function f via the recursion $x_{n+1} = f(x_n)$, with x_0 an arbitrary point in the domain. In this case, the sequence is called the orbit of x_0. To make sense, the range of f must be a subset of its domain.

The restriction to discrete systems is not as confining as it may seem at first glance. For instance, a great deal of the interest in chaotic dynamics has come from the study of phenomena modeled by systems of ordinary differential equations such as the systems of Lorenz, Rossler, Duffing: various systems for forced oscillations, etc. [Guckenheimer, 1983], [Drazin, 1992]. The modern interest in chaotic systems is sometimes dated from [Lorenz, 1963] (see the historical account in [Gleick, 1987]). Except in the rare cases for which explicit solutions are available, such systems are generally studied by examining an associated discrete system called the Poincare map [Guckenheimer, 1983], [Berge, 1984].

It is far beyond the scope of this work to give a comprehensive discussion of chaos. Indeed, there is no consensus in the literature as to exactly what the word means. One often finds general statements such as "Simply put, a chaotic system is a deterministic system that exhibits random behavior" [Parker, 1987]. While not wrong, it seems to us that such statements may be a bit misleading. For example, all random number generators in common use are actually pseudo-random, that is, deterministic algorithms designed for the very purpose of exhibiting behavior that seems random, and yet few would classify them as chaotic.

There are mathematical definitions of chaos for various types of systems, such as that of R. Devaney for an IFS [Devaney, 1989], but in fact such precise requirements can seldom be satisfied in a definitive way in an experimental

situation, where one is faced with the task of describing the behavior of the system as a whole based on a necessarily limited supply of data.

The approach to quantifying chaos that seems to have been most successful in experimental situations, and that is adopted here, is that of measuring dimensions of attractors. An attractor for an IFS is a set A that

(i) is invariant under f (i.e., $f(x) \in A$ for all $x \in A$; this says that once an orbit enters A, it never leaves), and

(ii) "traps" nearby orbits in the sense that if x_0 is sufficiently close to A, then the distance between the orbit of x_0 and the set A approaches zero.

Somewhat similar definitions can be given for attractors in systems that are not necessarily IFS's.

An attractor may be extremely simple. Consider, for example, the function $f(x) = x^2$, defined for all real numbers x. The one-point set {0} is an attractor: $0^2 = 0$, and if $-1 < x_0 < 1$, then $x_n \to 0$ as $n \to \infty$. On the other hand, an attractor for even a quite simple system may be extraordinarily complex. A well-known example is the Henon attractor that is discussed later.

The sorts of attractors that are commonly associated with chaos are the so-called strange attractors, that is those with non-integral dimension. Of course, one must carefully specify exactly what dimension means. Any reasonable *geometric* definition will classify points as zero-dimensional, curves as

one-dimensional, surfaces as two-dimensional, etc. The existence of mathematical objects with non-integral dimension has been recognized for about a century, but their significance for dynamics is more recent. Over the last several decades an enormous body of research has developed linking the existence of fractal attractors with chaotic dynamics, however defined.

The approach to dimension used is the correlation dimension (CD), proposed by Grassberger and Procaccia [Grassberger, 1983]. In contrast to purely geometric measures, like the Hausdorff dimension, the CD takes the dynamic behavior of a time series into account. For this reason, and also for reasons of computational efficiency, the CD has come into extensive use in recent years. In the survey [Ruelle, 1990] the author comments: "... this algorithm has played a very important role in allowing us to say something about systems that otherwise defied analysis." The next sections are devoted to the theory and implementation of the CD.

5.2 Dimension for Real Time Series

In this section, $X = x_1, x_2, \ldots$ is a sequence (time series) of real numbers. We assume that X is bounded, but some of the theory that follows can be adapted to unbounded sequences. Denote the finite sequence $x_1, x_2, \ldots x_N$ by X_N.

DEFINITION: For $r > 0$ and N a positive integer, let

$$C_N(r) = \frac{\#\ pairs\ from\ X_N\ for\ which\ |\ x_i - x_j|\ <\ r}{N^2} \qquad (5.1)$$

and define $$C(r) = \lim_{N\to\infty} C_N(r), \qquad (5.2)$$

if this limit exists. The pairs (i,j) are ordered, so there are exactly N^2 of them. The function C(r) is then the limiting mean number of pairs in the time series that lie within distance r.

In many cases of interest, C obeys a power law, that is; as $r \to 0$, $C(r) \sim r^{\upsilon}$, in the sense that for some constants $a > 0$ and $\upsilon \geq 0$,

$$\lim_{r\to 0} \frac{C(r)}{r^{\upsilon}} = a \qquad (5.3)$$

If this is so, the time series has CD υ. The general definition that follows is slightly broader.

DEFINITION: Suppose that C(r) is defined for sufficiently small r, and that

$$\lim_{r\to 0} \frac{\log(C(r))}{\log(r)} = \upsilon \qquad (5.4)$$

for some $\upsilon \geq 0$. Then the CD of X is υ.

In the appendix, it is shown that (5.3), which is often easier to work with, implies (5.4).

In order to clarify the meaning of this approach to dimension, here are a few elementary examples.

EXAMPLE 1: Suppose that the series is constant: $x_k = x$, for all k. Then from (5.3) it is easily seen that υ is 0. This agrees with our intuitive notion of dimension since a point is

a zero-dimensional object. Similarly, if X takes on only a finite number of values, for instance when it is periodic, $\upsilon = 0$.

EXAMPLE 2: Suppose that the series is purely random (i.e., independent and identically distributed) in [0,1]. It is easily seen that $C(r) = r$ for all $0 < r < 1$, and it follows that the CD is 1 here. Once again, this is reasonable: the random numbers "evenly fill" the one-dimensional object [0,1]. In the appendix it is shown that if X is evenly distributed in the weak sense that it has a bounded probability distribution on [0,1], then $\upsilon = 1$. (Incidentally the study of chaotic systems from the point of view of statistics and probability theory has become quite active in recent years. For surveys see [Berliner, 1992] and [Chatterjee, 1992].)

EXAMPLE 3: In this example, the elements of the time series are the binary rationals (i.e., the rationals whose denominators are powers of 2) in [0,1). We order them as follows:

$$0, \frac{1}{2}, \frac{1}{4}, \frac{3}{4}, \frac{1}{8}, \frac{3}{8}, \frac{5}{8}, \frac{7}{8}, \frac{1}{16}, \frac{3}{16}, \cdots \qquad (5.5)$$

Once again, the terms are "evenly spread" over [0, 1) and it is not difficult to see that $C(r) = r$, and thus that $\upsilon = 1$.

From the point of view of the CD, it appears that Examples 2 and 3 are quite similar, despite the fact that the binary rational sequence is very highly structured. Indeed, classical time series analysis using, say, the autocorrelation function, easily distinguishes between these kinds of

behaviors. We return to this point later.

Our next example serves to emphasize the dynamic, as opposed to geometric, nature of the CD.

EXAMPLE 4: Define a time series in [0,1] by $x_k = 1/2$ if k is odd, and x_k is random if k is even. The set of points $\{x_k \mid k = 1, 2, \ldots\}$ is identical to that of a random sequence in [0,1], but this time series is "concentrated" at 1/2, and the CD is 0. (This follows from the third lemma of the appendix, since it is clear that $C(r) \geq 1/4$ for all r here.)

EXAMPLE 5 (Cantor middle-halves set): An important fractal object is described here. For computational reasons, a middle-halves construction rather than the more familiar middle-thirds is used. It can be succinctly described by saying that one begins with the unit interval [0,1] and removes the open middle half, then remove the open middle halves of the resulting two intervals, etc., continuing this process indefinitely. What remains is the Cantor set H. To make this precise, define an operation R, "removal of open middle half" by

$$R[a,b] = \left[a, \frac{(3a+b)}{4}\right] \cup \left[\frac{(a+3b)}{4}, b\right] \qquad (5.6)$$

and generally, if $I_1, I_2, \ldots, I_m$ are pairwise disjoint closed intervals,

$$R\bigcup_{i=1}^{m} I_i = \bigcup_{i=1}^{m} RI_i \qquad (5.7)$$

Now let $H_0 = [0,1]$, and for $n \geq 1$, define $H_n = RH_{n-1}$. Finally,

define $$H = \bigcap_{n=1}^{\infty} H_n \tag{5.8}$$

Note that H_k consists of 2^k pairwise disjoint closed intervals, each of length 4^{-k}. Note also that the H_k are nested in the sense that $H_k \subset H_{k-1}$.

To get a CD a time series is needed. Here it is reasonable to take the new endpoints at each step in the generation of the H_k. This gives

$$0,\ 1,\ \frac{1}{4},\ \frac{3}{4},\ \frac{1}{16},\ \frac{3}{16},\ \frac{13}{16},\ \frac{15}{16},\ \ldots \tag{5.9}$$

We can guess the correct value of the CD by observing what happens when $r = 4^{-k}$. For example, if $r = 1/4$, it is true that $|x_i - x_j| \leq r$ precisely when both x_i and x_j lie in $[0,1/4]$ or both lie in $[3/4, 1]$; this happens half the time, so $C(1/4) = 1/2$. In general we see that $C(4^{-k}) = 2^{-k}$, so $C(r)/r^{1/2} = 1$ and we ought to have $\upsilon = 1/2$. Filling in details of the argument shows that this is indeed the case. Incidentally, the Hausdorff dimension of H is also 1/2.

EXAMPLE 6: To illustrate a few of the labyrinthine connections among fractals, dynamics, chaos, and computation, introduce the simple function,

$$f(x) = \begin{cases} 4x, & \text{if } -\infty < x \leq 1/2 \\ 4(1-x), & \text{if } 1/2 \leq x < \infty \end{cases} \tag{5.10}$$

which is plotted in Figure 5.1. We now consider the dynamics of the IFS generated by f. Start by noting that if $x_k < 0$ for some k, then $\lim_{n \to \infty} x_n = -\infty$. Next, if $x_k > 1$ for some k, then $x_{k+1} < 0$, and once again $\lim_{n \to \infty} x_n = -\infty$. Following this train of

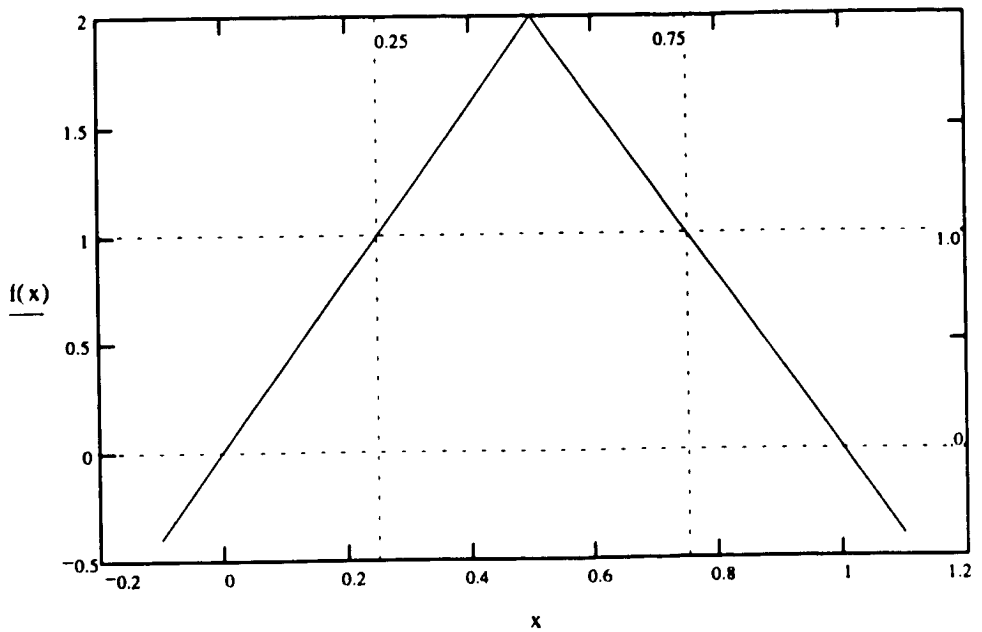


Figure 5.1 - Graph of equation (5.10).

thought, note that if $1/4 < x_k < 3/4$, we have $x_{k+1} > 1$, so the orbit diverges to $-\infty$. If $1/16 < x_k < 3/16$ or $13/16 < x_k < 15/16$, we get $1/4 < x_{k+1} < 3/4$, etc. At this point it should be clear that we are just generating H, and in fact one can prove that

(i) If $x_0 \in H$, all terms of the orbit remain in H, and

(ii) If $x_0 \notin H$, the orbit diverges to $-\infty$.

The orbits in case (i) are not analyzed, but we remark in passing that, by almost anybody's definition, there is chaotic behavior. Incidentally, *H* is not an attractor for this system. In fact, if $-\infty$ is regarded as an attracting "point", all points except those in H are in the attractor.

5.3 Correlation Dimension for Vector Time Series

In this section we look at time series $X = x_1, x_2, \ldots$ with terms in R^n, $n \geq 2$. The obvious reason for doing this is

that many processes (e.g. motion in three dimensional space) intrinsically give vector time series. There are other, less obvious, considerations also.

5.3.1 Correlation Dimension Definition

The CD for vector time series can be defined exactly as before, except that the absolute values in (5.1) are replaced by some vector "norm" (distance) $\|\cdot\|$. For example, if $\mathbf{v} = (v_1, v_2, \ldots, v_n)$ and $\mathbf{w} = (w_1, w_2, \ldots, w_n)$, use

$$\text{(i)} \quad \|\mathbf{v}-\mathbf{w}\| = \sqrt{\sum_{i=1}^{n} (v_i - w_i)^2} \tag{5.11}$$

or

$$\text{(ii)} \quad \|\mathbf{v}-\mathbf{w}\| = \sum_{i=1}^{n} |v_i - w_i| \tag{5.12}$$

or

$$\text{(iii)} \quad \|\mathbf{v}-\mathbf{w}\| = \max\{|v_i - w_i| \mid i = 1, 2, \ldots, n\} \tag{5.13}$$

If n = 2 or n = 3, the norm (i) is literally the distance between $\mathbf{v}$ and $\mathbf{w}$, and is called the Euclidean norm. The norm (iii) is the "maximum norm". For computational purposes, (ii) and (iii) are often convenient. Theoretically, it makes little difference which is used, since in a certain sense all norms on R^n are equivalent (See [Brock, 1986]).

The elementary properties of the CD given in the appendix for one-dimensional time series carry over to R^n with little difficulty. Some of them are listed here for convenience. It is assumed in (i), (ii), and (iii) that $X = x_1, x_2, \ldots$ is a bounded time series in R^n with a well-defined CD υ.

(i) $\upsilon \leq n$.

(ii) If $C(r) \leq kr^n$ for some constant $k > 0$ and all sufficiently small r, then $\upsilon = n$.

(iii) If X has a bounded probability distribution, then $\upsilon = n$.

(iv) If $C(r)/r^{\upsilon} \to a$ as $r \to 0$, for constants $a > 0$ and $\upsilon \in [0,n]$, then the CD is defined and is υ.

EXAMPLE 7: (random): If X is random (independent and identically distributed (i.i.d.), white noise, etc.) in some region of R^n, then $\upsilon = n$.

EXAMPLE 8: (Henon Attractor): This is one of the best-known strange attractors; see [Berge, 1984] and [Gulick, 1992]. Consider the function $F: R^2 \to R^2$ given by

$$F(x,y) = \begin{pmatrix} 1 - ax^2 + y \\ bx \end{pmatrix} \qquad (5.14)$$

where a and b are real numbers. Defining orbits as before by $\mathbf{x}_n = F(\mathbf{x}_{n-1})$ for $n \geq 2$, $\mathbf{x}_0$ arbitrary, it has been observed that for certain values of the parameters a and b, all orbits approach a complex set, the attractor, resembling an infinitely foliated boomerang. (See Figure 5.5.) On the attractors themselves, orbits bounce around in a chaotic fashion.

It seems virtually impossible to determine dimensions analytically here, so they must be estimated experimentally, i.e., by numerical calculation. We return to this point in the next section, but mention that it might be expected that the result is strictly between 1 and 2.

5.3.2 High-Dimensional Systems

The examples that have been considered so far have been rather simple, in the sense that we have in each case been able to give explicitly a generating mechanism for the time series under consideration. Far more common for the experimentalist is the situation in which the generating mechanism is either unknown or difficult to describe analytically. Also typical is to have a low-dimensional (often, one-dimensional) time series generated by a high-dimensional process. The problem here is to capture, somehow, the essence of the high-dimensional process using low-dimensional evidence. Imagine, for example, trying to describe the motion of a 3-body system under gravitational attraction, given only the momentum in the x-direction of one of the particles measured at one-second intervals.

To narrow things a bit, suppose we wish to establish the existence or non-existence of a low-dimensional attractor in a high-dimensional system, using only a time series x_1, x_2, ... of real numbers. What is done in practice is to attempt an analysis of "vectorized" time series of form

$$(x_i, x_{i+1}, \ldots, x_{i+m-1}) \qquad (5.15)$$

for various vector lengths m. This is an "embedding" of the data into R^m. To give a rationale for doing this, we take a short theoretical detour.

Let us restrict our attention for the time being to iterated function systems (IFS). Suppose that we have a function

$$f: S_1 \rightarrow S_1, \tag{5.16}$$

for some set S_1. It is often the case that the dynamics of f are difficult to analyze directly, or that no explicit expression for f is available at all. One strategy for studying the system (5.16) is to model it with some other, more tractable, IFS

$$g: S_2 \rightarrow S_2. \tag{5.17}$$

We would like the dynamics of g on S to mimic or "code" the dynamics of f on S_1, and it is this notion that we must make precise.

An obvious requirement is that S_1 and S_2 be the same size, in the sense that there is a 1:1 correspondence between them. Call the correspondence function $\varphi: S_1 \rightarrow S_2$, and think of φ as a sort of translation of S_1 into S_2. Now suppose that x_0 is any element of S_1. Let y_0 be the "twin" of x_0 in S_2, i.e., $y_0 = \varphi(x_0)$. For the orbit of x_0 to translate into that of y_0, we need y_1 to be the "twin" of x_1, i.e., $y_1 = \varphi(x_1)$. This leads to the requirement that $g(\varphi(x)) = \varphi(f(x))$ for all x in S_1.

The situation can be visualized with the diagram shown in Figure 5.2. This is what is called a "commutative diagram"; points in the upper left arrive at the same destination in the lower right whether they go right then down, or down then right.

Finally, orbits of g truly behave like those of f only if the correspondence somehow preserves a notion of distance. A minimal requirement is that φ and its inverse φ^{-1}

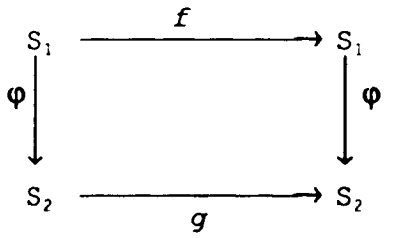


Figure 5.2 - Commutative diagram.

be continuous. (Of course, the continuity depends on the structures of S_1 and S_2. We deal mainly with subsets of R^k, where continuity has the normal meaning given in calculus.) A 1:1 correspondence that is continuous is called a homeomorphism. The needed definition can now be made.

DEFINITION: The IFS f: $S_1 \rightarrow S_1$ is equivalent to the IFS g: $S_2 \rightarrow S_2$ if there is a homeomorphism φ: $S_1 \rightarrow S_2$ such that $\varphi(f(x)) = g(\varphi(x))$ for all x in S_1.

We illustrate the utility of these ideas by a simple but interesting example (see [Blaine, 1991], [Gulick, 1992], [Shaw, 1981]). Let $S_1 = [0,1)$ and consider the function $f(x) = 2x \pmod 1$; i.e., $f(x) = 2x$, if $0 \leq x < 1/2$, and $f(x) = 2x - 1$, if $1/2 \leq x < 1$. The nature of f suggests that we work in binary, and it is easy to see that the action of f is to move the point in the binary representation of a number in [0,1) one place to the right, and then remove the whole-number part. Now, let S_2 be the set of infinite sequences with terms in {0,1} that do not end with an infinite string of 1's. The homeomorphism φ is defined by assigning to each number x in [0,1) the sequence given by its binary expansion. (The stipulation on infinite strings of 1 is to ensure uniqueness.

For example, 1/2 can be represented as .1 or as .011111.....; the sequence assigned is 1,0,0,0, ... rather than 0,1,1,1, ...) Let g: $S_2 \to S_2$ be the operation "erase leftmost term". The technicalities on continuity are omitted here. It may seem that things merely have been relabeled but note that the dynamics of g are quite transparent. For example, a sequence that, perhaps after a finite number of initial terms, consists of a repeating block has an orbit that is eventually periodic. Such sequences correspond precisely to rational numbers in [0,1), and it can be concluded that f has infinitely many periodic orbits (of every possible period length), corresponding to the rationals, and infinitely many non-periodic orbits, corresponding to the irrationals.

Now we are ready to discuss the use of (5.15). The theory is due to F. Takens (see [Takens, 1983], [Takens, 1980]); we give only a very brief summary.

Suppose that we are studying an IFS F: $R^n \to R^n$, but that the structure of F is completely hidden from our view. (The dimension n may not even be known.) What is assumed is that there is a one-dimensional measurement h of some sort, so that instead of orbits $\mathbf{x}_1$, $\mathbf{x}_2$, ... of F in R^n, we actually observe sequences $h(\mathbf{x}_1)$, $h(\mathbf{x}_2)$, ... The task is to deduce the true behavior of F from the one-dimensional evidence.

Writing $x_i = h(\mathbf{x}_i)$, expression (5.15) can be written in the form $(h(\mathbf{x}_1), h(F(\mathbf{x}_1)), \ldots, h(F^{m-1}(\mathbf{x}_1)))$. This leads us to define a function J: $R^n \to R^m$ by

$$J(\boldsymbol{x}) = \left(h(\boldsymbol{x}),\ h(F(\boldsymbol{x})),\ \ldots,\ h(F^{m-1}(\boldsymbol{x}))\right). \qquad (5.18)$$

Takens' result asserts that, under some mild smoothness assumptions on h and F, the function J is "generically" a diffeomorphism (i.e., a differentiable homeomorphism) for all sufficiently large m. (The word generic means roughly that the exceptions are isolated.) Note that it is not asserted that the range of J is all of R^m; if we denote the range of J by Ω, then J is, generically, a diffeomorphism $R^n \rightarrow \Omega$.

The import of this is that the observed time series (5.15) may indeed mimic the behavior of the IFS F. The mathematics is quite complicated, but there is a fairly clear intuitive interpretation. Imagine that F generates an attractor, possibly of non-integral dimension. The attractor in the time series (5.15) could not be expected to be "seen" unless m were large enough to "embed" it.

5.4 Practical Computation of the Correlation Dimension

How does one actually go about making a numerical estimate of the CD of a given system? Two obvious, but quite important, considerations are:

(i) only a finite amount of data is ever available, and

(ii) data is usually subject to measurement error of some sort, such as error created by the limited precision of a physical measuring device, or round-off error due to the finite word length on a computer.

There needs to be enough data with enough precision so

that C_N is a good approximation to C, and so that we can get a good estimate of $\lim_{r \to 0} \log(C(r)) / \log(r)$ by taking small values of r.

Faced with the task of estimating a CD from a truncated time series $X_N = x_1, x_2, \ldots, x_N$, we essentially run through pairs and measures distances. For a given value of r, a tally is kept of the number of pairs with distance less than r, and when done, we have $C_N(r)$. In practice, many values of *r* will be used. (After all, we don't know in advance how small a value is needed.) With this in mind, we can tally the number of distances in various categories or "bins"; typically we might use bins of type $[2^{-k-1}, 2^{-k})$ for some range of integers *k*. There are efficient algorithms for doing this [Parker, 1987], and this efficiency is one of the things that makes the CD, as opposed to other measures of dimension, so attractive a tool.

Now, if C(r) has been accurately estimated and *r* is small, the CD is well approximated by the ratio log(C(r))/log(r). In particular, if this ratio remains nearly constant over several orders of magnitude of small r, we can be reasonably confident that the estimate is meaningful. For this reason, a graph of log(C(r)) vs. log(r), using linear interpolation between the calculated points, is useful; the slope, if nearly constant over a good interval of the horizontal axis, is the approximation to the CD.

There is one serious consideration about the size of *r* that must be kept in mind. As mentioned above, we are usually

working with data values given only with a certain precision, say $\pm\eta$. If r is the same order of magnitude as η, or less, then we expect that the results would be contaminated. To illustrate with a very crude example, suppose that our time series consists of terms in [0,1) and that only the first two digits of each term are known to be correct. If the remaining digits are truncated to 0, and r is taken to be less than .01, we "see" only 100 points at most, and the CD is estimated to be 0. If, on the other hand, the remaining digits consist of random values (white noise), and r is much smaller than .01, we only "see" the noise, and the CD is estimated to be 1.

For the above reasons, we look for consistent slope somewhere in the middle regions of the graph. Of course, we are seldom absolutely sure where good values stop and contamination begins - this is an inescapable fact of life for the experimentalist.

When working with time series that consist of one-dimensional measurements on a large-dimensional process, as described above, the usual procedure is to perform the computations on the original data $x_1, x_2, \ldots$, and also on vector data $(x_1, x_2, \ldots, x_m), (x_2, x_3, \ldots, x_{m+1}), \ldots$, for various vector lengths m. When the graphs are constructed and one finds consistent slopes for values of m above a certain threshold, there is strong evidence of an attractor in the high-dimensional process. Some elementary examples are given in the next section to clarify this.

In [Ruelle, 1990], the author points out that a

calculated CD υ based on a finite time series X_N of N terms should satisfy the inequality

$$\upsilon < 2\log(N) \tag{5.19}$$

where the base 10 logarithm is used, and where it is assumed that the slope is measured over at least one decade. Some trenchant comments follow concerning purported CD calculations in the literature that violate (5.19).

The argument for (5.19) is quite simple. Suppose that $r_1 < r_2$. The approximating slope is

$$\frac{\log(C_N(r_2)) - \log(C_N(r_1))}{\log(r_2) - \log(r_1)} \tag{5.20}$$

Now let $M_N(r) = N^2 \cdot C_N(r)$; i.e., $M_N(r)$ is the number of pairs of terms with distance less than r. The numerator in (5.19) is equal to $\log(M_N(r_2)) - \log(M_N(r_1))$.

Now, $M_N(r_1) \geq 1$ and $M_N(r_2) \leq N^2$. The requirement that slope is measured over at least one decade means precisely that $r_2 \geq 10r_1$. It follows that $\log(M_N(r_2)) - \log(M_N(r_1)) \leq 2\log(N)$, and $\log(r_2) - \log(r_1) \geq 1$. The result follows immediately. (It should be noted that the use of base 10 is not essential. Ruelle argues, however, that measurements over much less than one decade may be unreliable.)

Finally, note that, unfortunately, there is still a certain amount of ambiguity in the interpretation of some CD calculations. Brock states it well: "In practice a range of ε's over which the slope of the plot appears to be 'stable' is chosen by 'eyeballing'." [Brock, 1986]

The implementation of our CD algorithm may be demonstrated now. The basis for this algorithm is from [Parker, 1987]. As mentioned earlier, an efficient algorithm for calculating the CD involves computing the distance between pairs of vectors and then counting the number of vectors at that distance. As can be seen from the pseudocode below, the vectors $\mathbf{v}_i = x_i, x_{i+1}, \ldots, x_{i+d-1}$ and $\mathbf{w}_i = x_{i+d}, x_{i+1+d}, \ldots, x_{i+2d-1}$ are formed from the time series data $X_N = x_1, x_2, \ldots, x_N$. Distances between the vectors are calculated using equation (5.11).

The integer part of $\log_2$ of the distance calculated above forms the index (P) to an array. The distances are then accumulated by incrementing the exponent-indexed array element. The variables P_{max} and P_{min} capture the largest and smallest values of the exponent. P_{max} and P_{min} form the x-axis limits of the CD plot. CD plots are shifted 2^{23} to the right. This yields positive numbers for $\log_2(r)$.

This algorithm is repeated for all embedding dimensions under consideration. The exponent array is reset (i.e., all values equal 0) for each dimension. The pseudocode for the algorithm is shown in Figure 5.3.

The following example illustrates the use of the algorithm.

EXAMPLE 1: Suppose that X is random (i.i.d.) in some interval [a,b]. We have already seen that the CD is 1. If we calculate with (5.15), we are essentially generating random numbers in R^m, and so the CD will be m.

```
begin
  for d = 1 to d_MAX do        { d_MAX = max embedding dimension}
  begin
    reset exponent array;      { elements = 0 }
    M := N DIV d;              { M = number of data vectors }
    for i = 1 to M-1 do
      for j := i+1 to M do
      begin
        summ := max { |v_i - w_j|, minimum };
        { i.e., the distance as given by (5.11) }
        { ... and minimum = 2^-21 }
        summ := SQRT(summ);
        P := TRUNC(log2(summ));        { integer portion }
        P := P + offset;               { so that P ≥ 0 }
        Inc(expo[P]);                  { count powers of 2 }
        if (P > P_max) then P_max := P;
        if (P < P_min) then P_min := P; { max & min expo's }
      end;  { vectors }
    for x_value := P_min to P_max do
    begin
      sum       := sum + expo[x_value];
      y_value   := log2( sum / (N·(N-1)));
      Plot(x_value, y_value);
    end;
  end;  { dimensions }
```

Figure 5.3 - CD calculation Algorithm.

The import of this example is simple but important. Figure 5.4 is generated from 2048 one-dimensional data values (from a pseudo-random number generator). When the CD of this data is calculated, we observe an increase in slope of approximately 1.0 with increasing embedding dimension. We should suspect that there is no low-dimensional attractor; i.e., that we are observing a random process.

EXAMPLE 2: Let us revisit the binary data of expression (5.5). The CD of this data is 1. If any embedding dimension $m \geq 2$ is chosen, the CD is still 1.

The CD plot of the binary fractions is shown in Figure 5.5. The slope of each embedding dimension is found using the

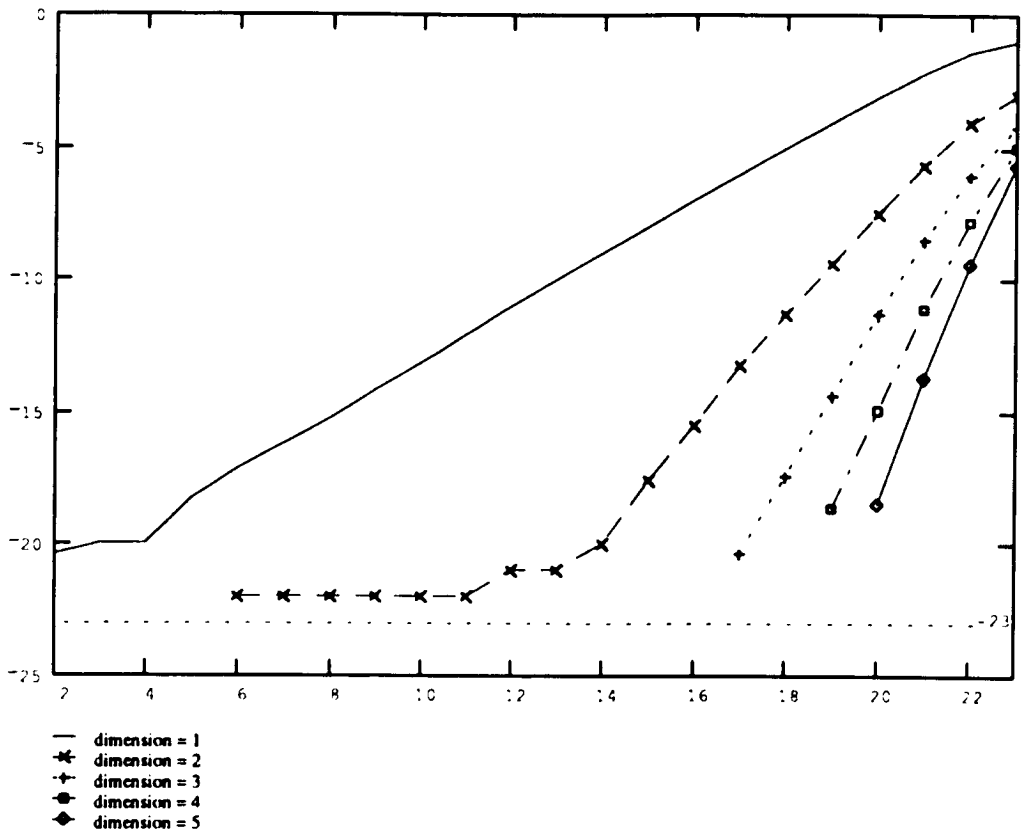


Figure 5.4 - CD plot of random data: $\log_2 C_x(r)$ vs. $\log_2(r)$.

right-hand portion of the graph. Linear regression was used to calculate the slope of each line. "Avg_slope" is the average of the slopes of embedding dimensions 2, 3 and 4. These dimensions give consistent slopes. This is due to the smoothing effect of higher dimensions on one hand and the limitation of vector length versus the number of data points on the other.

We outline the reason for the behavior of the CD of EXAMPLE 2 for m = 2. The general case is similar. Group the data as follows:

(1) (0, 1/2) (1/4,3/4)

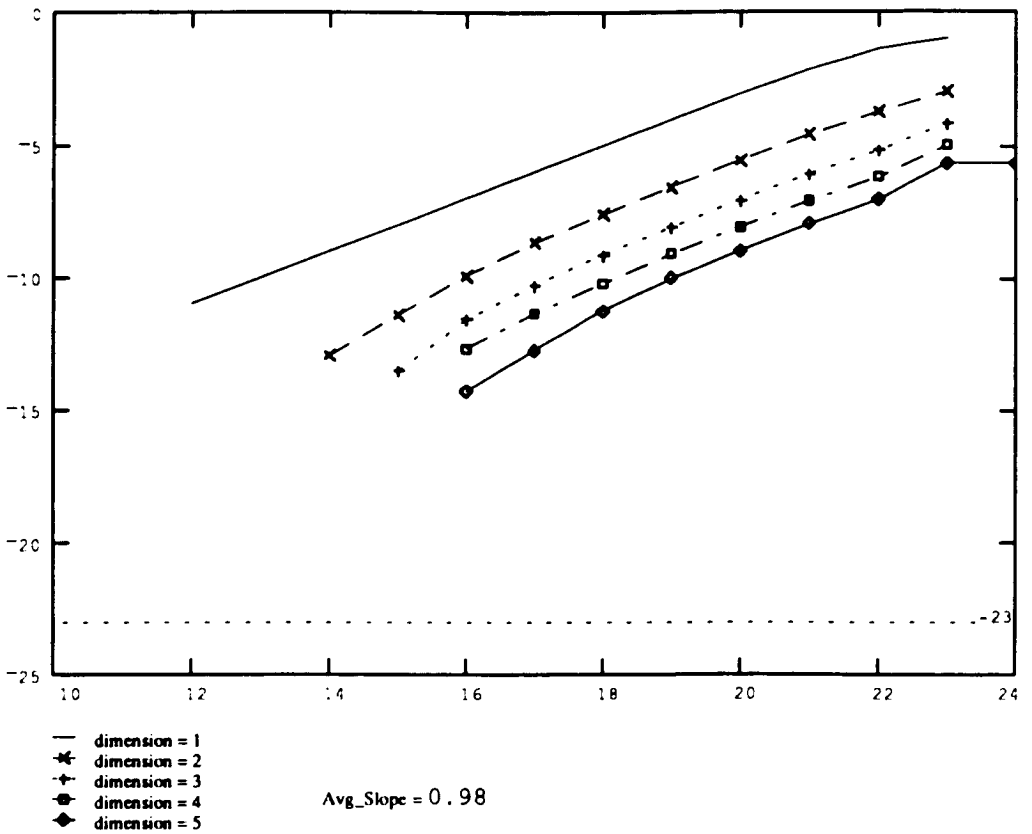


Figure 5.5 - CD plot of binary fractions: $\log_2 C_N(r)$ vs. $\log_2(r)$.

(2) (1/8,3/8), (5/8,7/8)

(3) (1/16,3/16), (5/16,7/16), (9/16,11/16), (13/16,15/16)

etc.

Now note that the data in group (k) all lies on the line $y = x + 2^{-k}$. In other words, the (one-dimensional!) line y = x acts as an attractor. This data, in contrast to the random data in the previous example, is intrinsically one-dimensional, and this is reflected by the CD.

Binary data is used in this example simply because there is no round-off error in its representation in a computer.

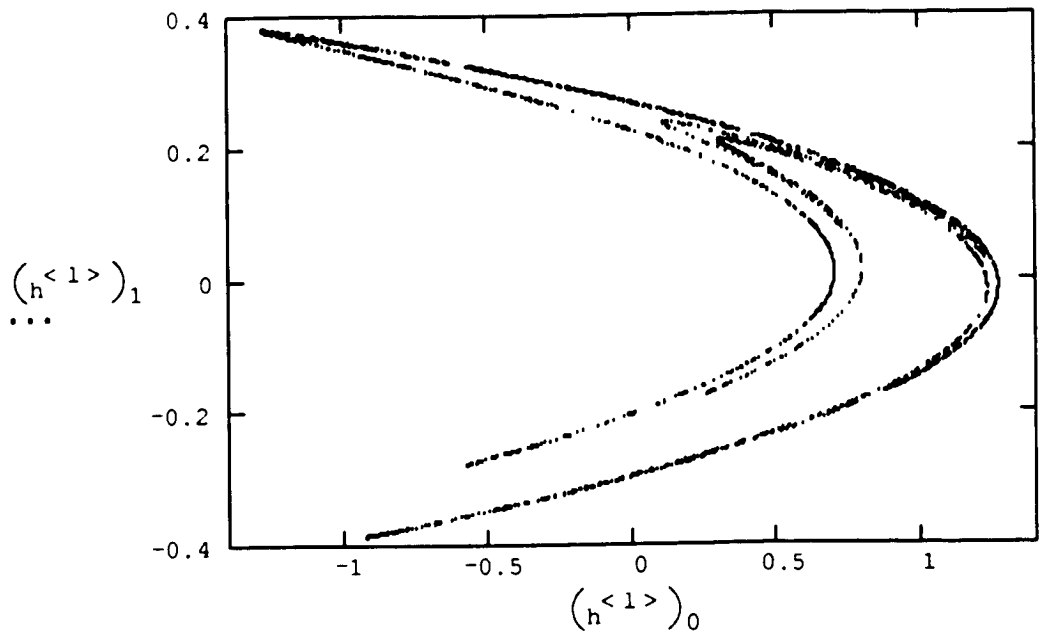


Figure 5.6 - Henon Attractor.

EXAMPLE 3: Here is the promised calculation for the Henon map (Equation 5.14). The parameters are set at a = 1.4, b = 0.3, and the initial point is (0.05, 0.12). To reduce the effects of transient behavior, 100 have been performed iterations before recording data. Figure 5.6 gives some idea of the structure of the attractor.

One purpose of presenting this example is to illustrate the embedding process. To that end, we have chosen the very simple measurement function h(x,y) = x, i.e., just take the x coordinate of each point. The results are given in Figure 5.7. The CD is estimated by linear regression at 1.21. The average slope of the 2nd through 4th embedding dimensions is shown in the figure.

EXAMPLE 4: Figure 5.8 presents the results from the Cantor set data (5.9). As expected, linear regression yields a slope

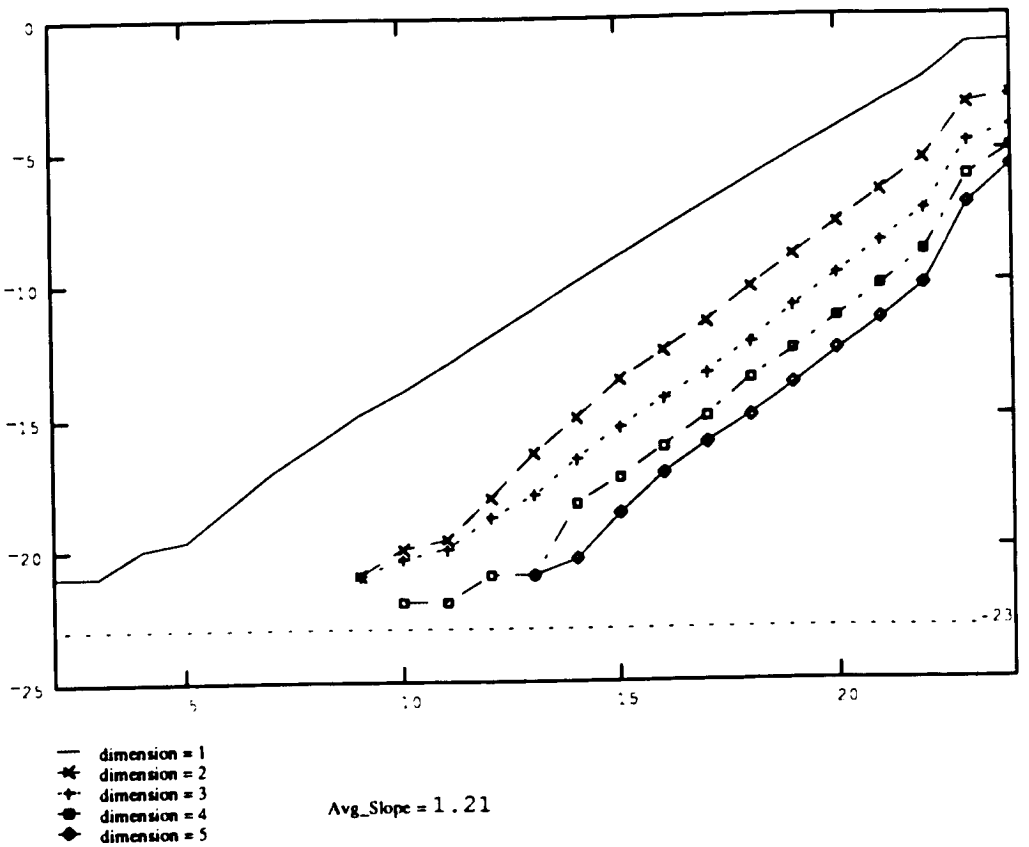


Figure 5.7 - CD plot of Henon attractor: $\log_2 C_N(r)$ vs. $\log_2(r)$.

very close to 0.5 for every embedding dimension used. Once again, the binary nature of the data eliminates round-off error.

5.5 Periodicity and Randomness in Dynamical Systems

So far we have considered a certain sort of structure on time series, namely a dimensional structure via the CD. The impetus for this sort of analysis comes from the fact that fractal structures are of great importance in the study of dynamical systems. The extremes in the behavior of the CD of

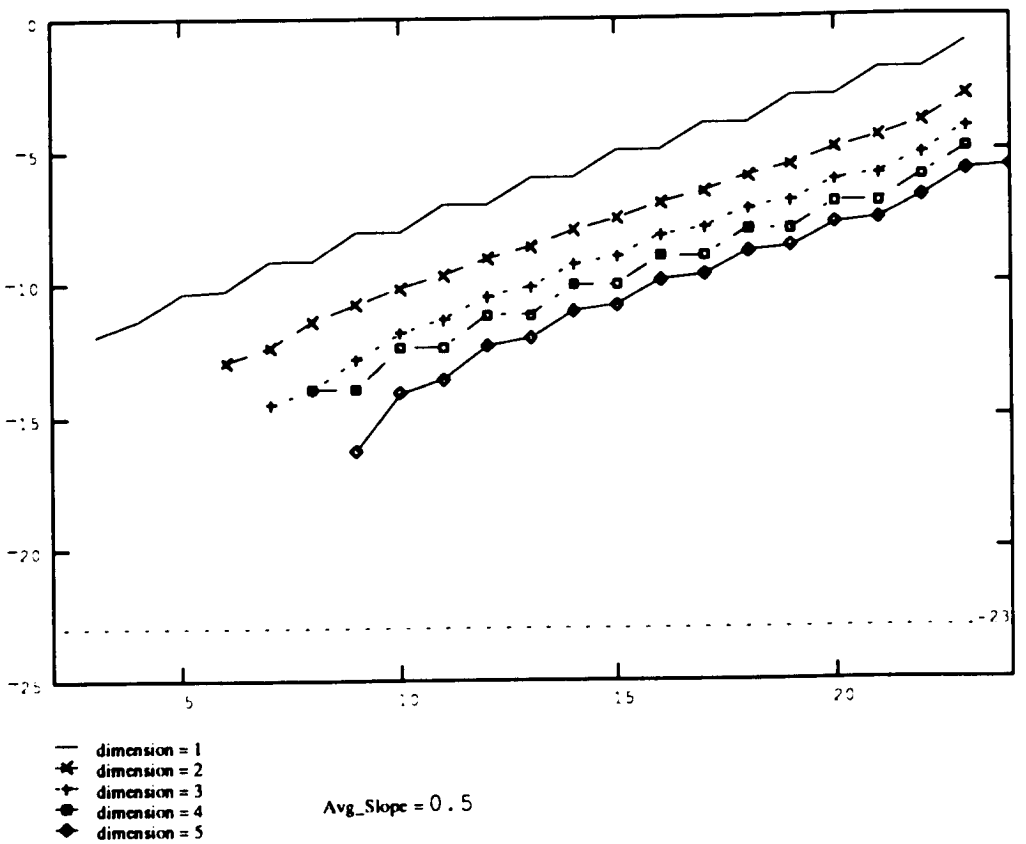


Figure 5.8 - CD plot of Cantor set: $\log_2 C_N(r)$ vs. $\log_2(r)$.

time series occur when it is constant or periodic (CD = 0), or when it is purely random (CD = n, where the series lies in R^n). In this section we take a closer look at the important issues of periodicity and randomness in dynamical systems.

First, consider questions of periodicity. Periodic behavior is, in a sense, as non-chaotic as it is possible to be. Quasi-periodicity is defined as the superposition of two or more periodic signals with (possibly) incommensurable periods; i.e., the ratios of the frequencies may be irrational. Quasiperiodic signals can be highly irregular, but are not considered chaotic by most definitions [Grassberger,

1983], [Parker, 1987].

We must be careful about the terminology here. A time series is periodic with period m, if it repeats itself every m terms; i.e., $x_{i+m} = x_i$ for all i. Strictly speaking, there is no such thing as a quasiperiodic time series; if X has period m_1 and Y has period m_2, then X + Y has period m, where m is the least common multiple of m_1 and m_2. When we refer to "periodic behavior" of a time series, we include the possibility that it is generated by sampling a periodic process at regular intervals which do not necessarily stand in rational relation to the period of the process. Quasiperiodic behavior refers to superpositions of periodic behaviors. Now, the CD of a periodic time series is 0, but if a time series has periodic behavior in the sense just mentioned, it may be difficult to make any sense at all of the results of a CD computation. Our next examples explore these themes.

EXAMPLE: Analyze 2048 terms of a time series whose first few terms are, to five significant digits,

1.6282, -0.0620, 0.5535, -0.2946, -1.9419, 0.4719, 0.7124, 0.1696,
1.3687, -.9052, -1.5107, 0.4551, -0.2934, 0.8799, 1.5006, -1.2269, ...

To the naked eye, this data appears rather random, or perhaps chaotic in some sense. However, our CD algorithm gives the results shown in Figure 5.9, which seem to defy interpretation. It seems that the CD is simply the wrong tool here. If this data is run through a standard Fast Fourier Transform algorithm, Figure 5.10 is obtained. These results

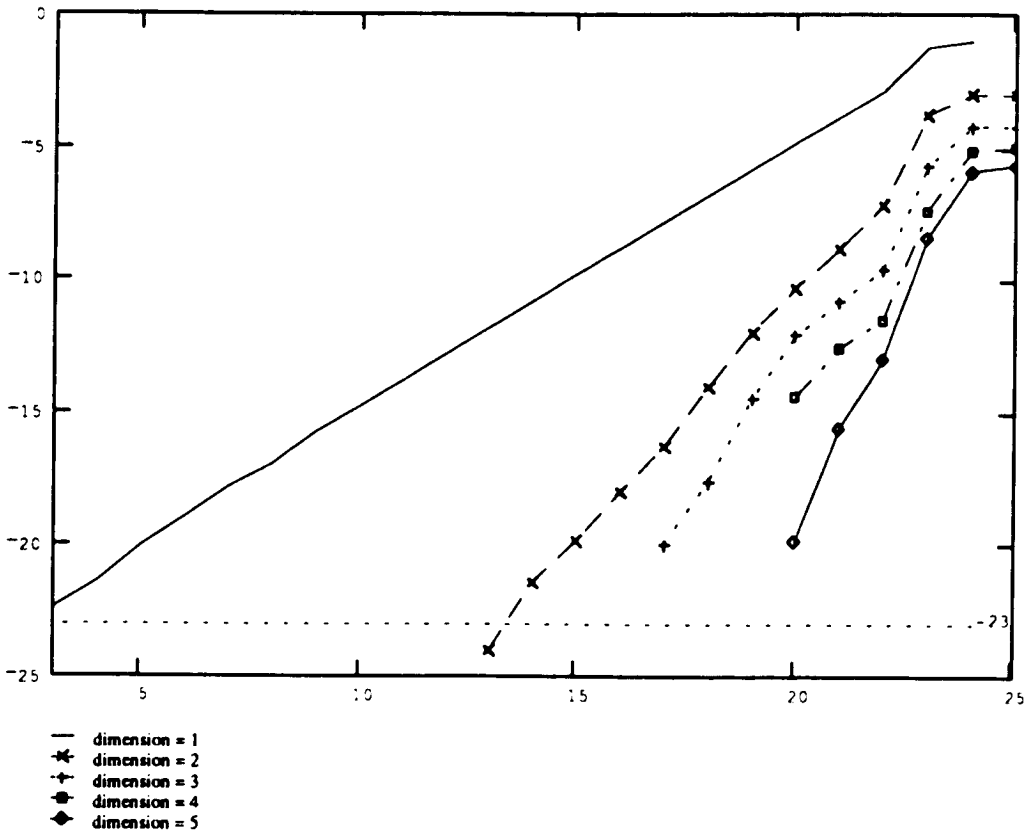


Figure 5.9 - CD plot of time series: $\log_2 C_N(r)$ vs. $\log_2(r)$.

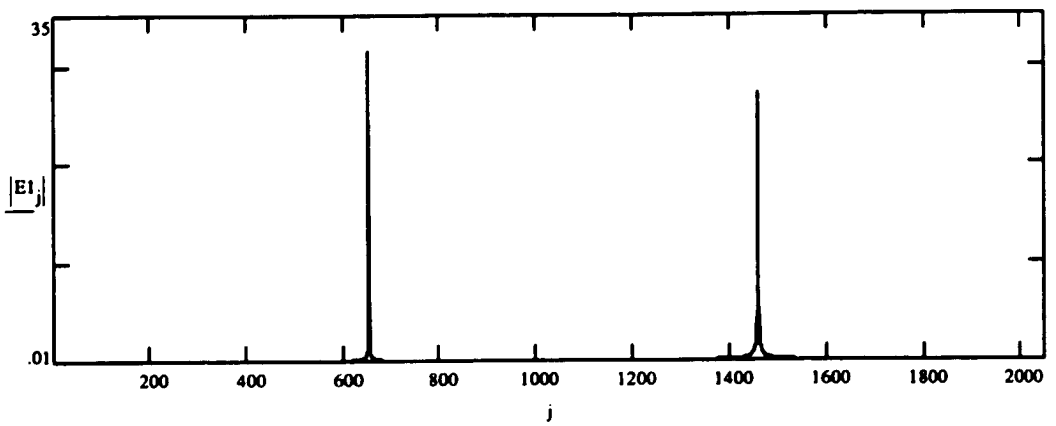


Figure 5.10 - FFT of time series.

indicate that we have quasiperiodic behavior with two dominating frequencies. The first spike is at approximately

652 while the second is at 1458. Scaling these results by the FFT input data length yields $f_1 = 0.159$ and $f_2 = 0.356$. This is in fact correct. The time series in question is

$$x_n = \sin(n) + \sin(\sqrt{5}\, n) \tag{5.21}$$

Here we make a few comments on the use of Fourier analysis in the study of dynamical systems. For the detection of quasiperiodic behavior, the Fourier transform is the best thing to use; indeed, this is the very purpose of Fourier analysis. From the point of view of quantifying chaos, then, the Fourier transform can be regarded as a negative indicator; if the signal is clearly quasiperiodic, it is not chaotic. The following statements are found in the literature:

> "... when observing a seemingly strange behavior, one would like to have clear-cut procedures which could exclude that the attractor is indeed multiply periodic, ... [this] possibility can be ruled out by making a Fourier analysis ... " [Grassberger, 1983]

What else the Fourier Transform can tell us about data from dynamical systems is not entirely clear. The literature is replete with rather general assertions about the broad-band nature of spectra of chaotic signals. For example,

> "... the broad-band noise characteristic of chaos should start to appear." [Berge, 1984]

> "Though a broad spectrum does not guarantee

sensitivity to initial conditions, it is, in practice, a reliable indicator of chaos." [Baker, 1990]

"One of the clues to detecting chaotic vibration is the appearance of a broad spectrum of frequencies in the output ... "[Moon, 1992]

We must use considerable caution in applying such criteria, however. Consider the Fourier spectra shown in Figures 5.11, 5.12 and 5.13. Two of them were calculated using data from well-known IFS's that are universally regarded as chaotic, while the remaining one comes from random number data.

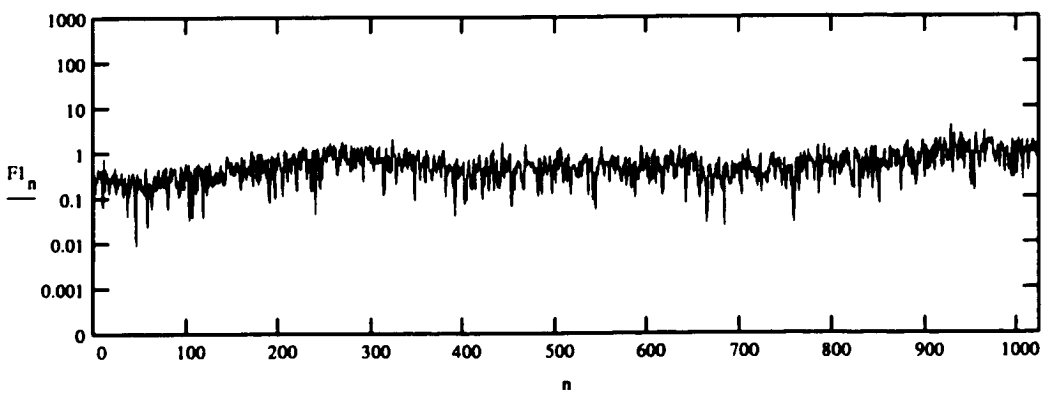


Figure 5.11 - FFT "A".

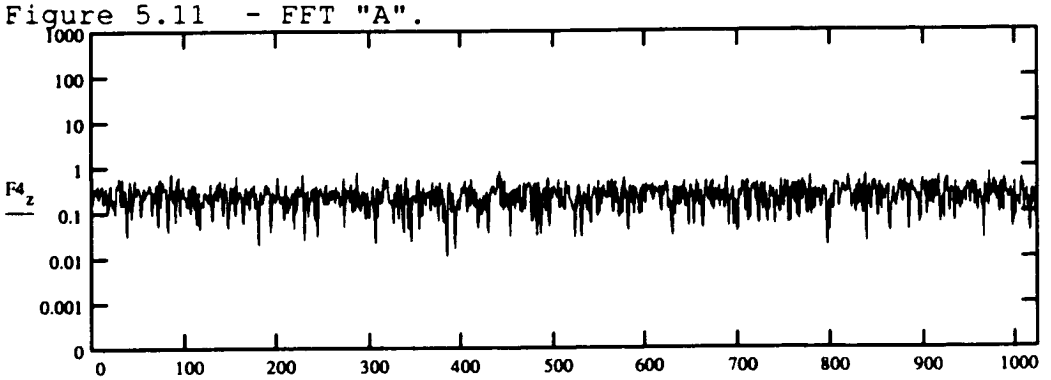


Figure 5.12 - FFT "B".

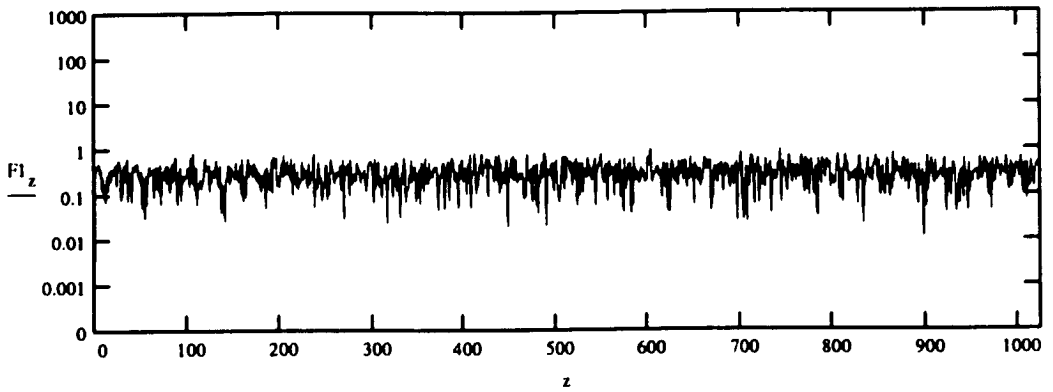


Figure 5.13 - FFT "C".

Without the labels, it would be quite difficult to tell which is which. The spectra are identified below.

FFT "A": Data from Henon map with parameters a = 1.4, b = 0.3, and initial point (0.05, 0.12). To reduce effects of transient behavior, points x_{100} through x_{2047} were used. (Figure 5.11)

FFT "B": Data consists of 2048 points of random (i.i.d.) numbers in [0,1]. (Figure 5.12)

FFT "C": Data from IFS defined by the quadratic map f(x) = 4x(1 - x), with domain [0,1]. Initial point was 0.70, 2048 points used. (Figure 5.13)

There seems to be no widely applicable method for distinguishing Fourier spectra of noise from Fourier spectra of chaotic signals. This reinforces our use of Fourier analysis solely as a means of detecting periodic and quasi-periodic behavior. (It is true that a sequence of period-doubling bifurcations - a possible route to chaos in some systems dependent on a parameter - has a recognizable

signature in the Fourier spectrum. For treatments, see [Baker, 1990], [Crutchfield, 1980], [Fenstermacher, 1979], [Gollub, 1980]. This phenomenon was not encountered in the experiments conducted on 25 and 400 processor meshes [Ilyadis, 1990].)

5.6 Measurements of Experimental Data

In this final section some further ideas on the interpretation of CD algorithms, especially in connection with "noisy" data are explored.

It is instructive to compare the CD once again with the Fourier transform. One of the most useful properties of the latter is its linearity: the transform of the superposition of two signals is the superposition of their transforms. In particular, a periodic or quasi-periodic signal contaminated by the superposition of random noise has a Fourier transform showing a distinct spectral structure rising above "clutter" induced by the noise. The CD has no such property, and interpretation of the output of the algorithm can be somewhat delicate.

To quickly summarize the interpretation of our CD algorithm so far, we look at a range of values of r for which the plot of log(C(r)) vs. log(r) has positive slope, and try to find within this range some evidence that the slopes are nearly constant over some subrange. If this is the case, we have an estimate for the CD. There is some qualitative evidence, at least, that under certain conditions we may be

able to glean some additional information from the output.

5.6.1 Data from the Quadratic Map

Take as a "canonical" example the extensively studied IFS generated by the quadratic map (also called the logistic equation)

$$f(x) = \lambda x(1 - x). \tag{5.22}$$

We compare the output of the CD algorithm for some "noisy" versions of (5.22) with the output for our simulator and attempt an interpretation. It should be emphasized from the outset that the results, here and in the literature to date, are strongly suggestive but not rigorous. Berliner comments

> "Even in very simple examples, such as the logistic map observed with independent Gaussian errors, the resulting likelihood functions can be extremely complex, intractable objects." [Berliner 1992]

To set the stage, the results for (5.22) with λ = 3.5699456 (Figure 5.14) and λ = 4.0 (Figure 5.15) are presented. What we get is very clean, and consistent with [Grassberger, 1983].

Now contaminate (5.22) with noise, by which we always mean white noise. (Incidentally, noise in experimental data may truly be contamination, or may be due to some stochastic element in the system itself (see equation (1.1). For the purposes of our analysis, it does not matter which.) Three ways of doing this are illustrated. Here assume that $@_0$, $@_1$, ... is i.i.d. in [0,1).

(i) Interspersed uncontaminated data with noise every

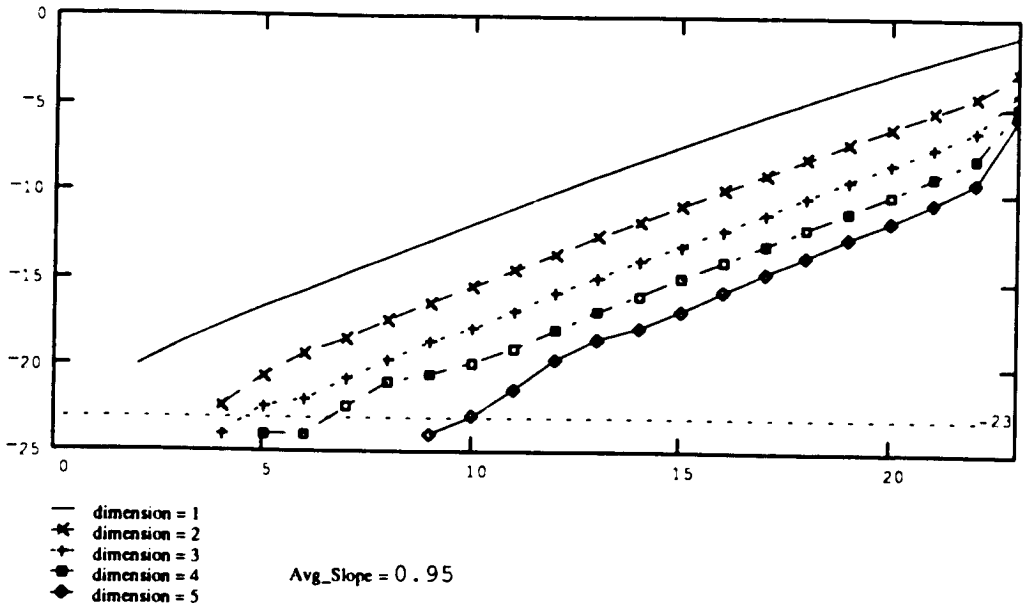


Figure 5.14 - CD plot of quadratic map with $\lambda = 4.0$: $\log_2 C_N(r)$ vs. $\log_2(r)$.

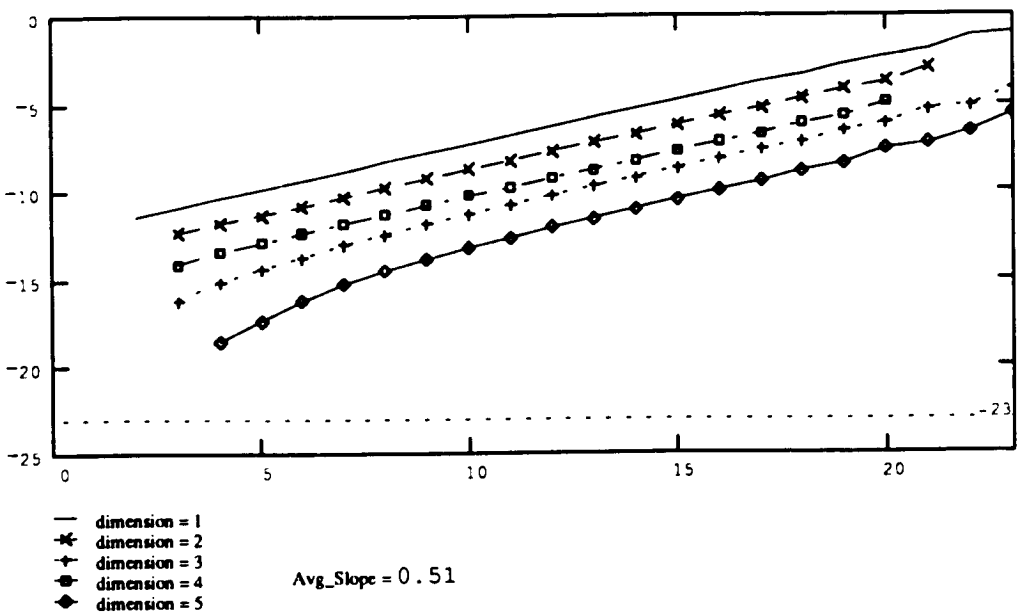


Figure 5.15 - CD plot of quadratic map with $\lambda = 3.5699456$: $\log_2 C_N(r)$ vs. $\log_2(r)$.

10 terms to get x_0, x_1, ... , x_8, $@_0$, x_{10} ...

(ii) Superimpose noise on uncontaminated data to get

$x_0 + 0.1 \cdot @_0$, $x_1 + 0.1 \cdot @_1$, ...

(iii) Introduce noise into the evaluation of the function to get a sequence of form $x_n = f(x_{n-1} + (0.01 \cdot @_n))$. Here we must add the proviso that if $x_n < 0$ or $x_n > 1$, then $x_n = 0$.

The results are shown in Figures 5.16, 5.17 and 5.18.

What is striking here is that if we measure slopes over two different ranges of r, we see that over one of them, the

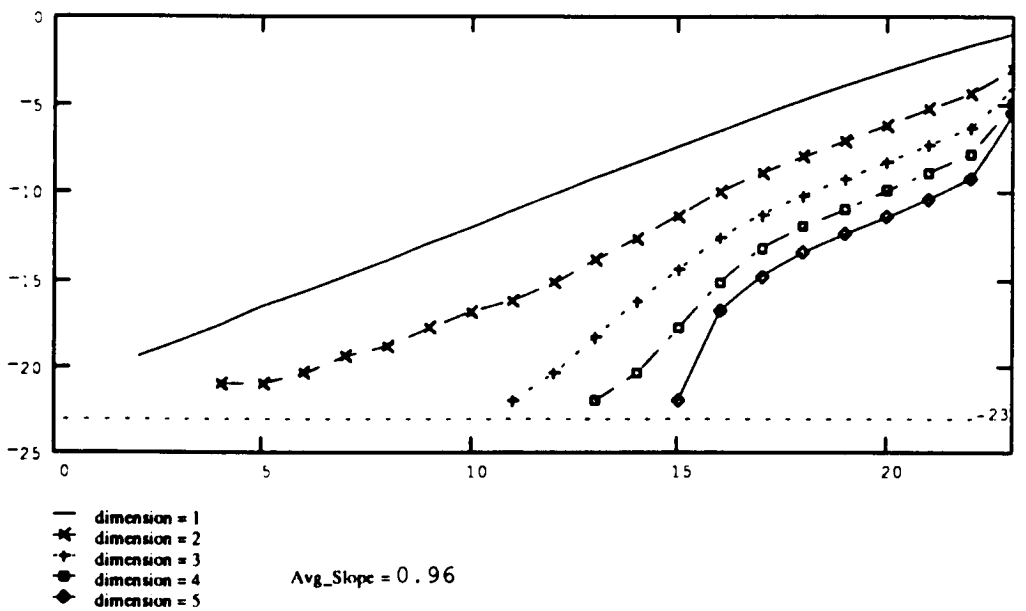


Figure 5.16 - CD plot of quadratic map with noise introduced into function evaluation: $\log_2 C_N(r)$ vs. $\log_2(r)$.

results are definitely behaving as random data (slope ~ embedding dimension m), and over the other, the slopes are roughly constant. This suggests that the CD algorithm can separate randomness and dimensional behavior. Some qualitative arguments along these lines, and some suggestions toward using results like these to quantify the amount of noise in a system may be found in [Ben-Mizrachi, 1984] and [Berliner, 1991].

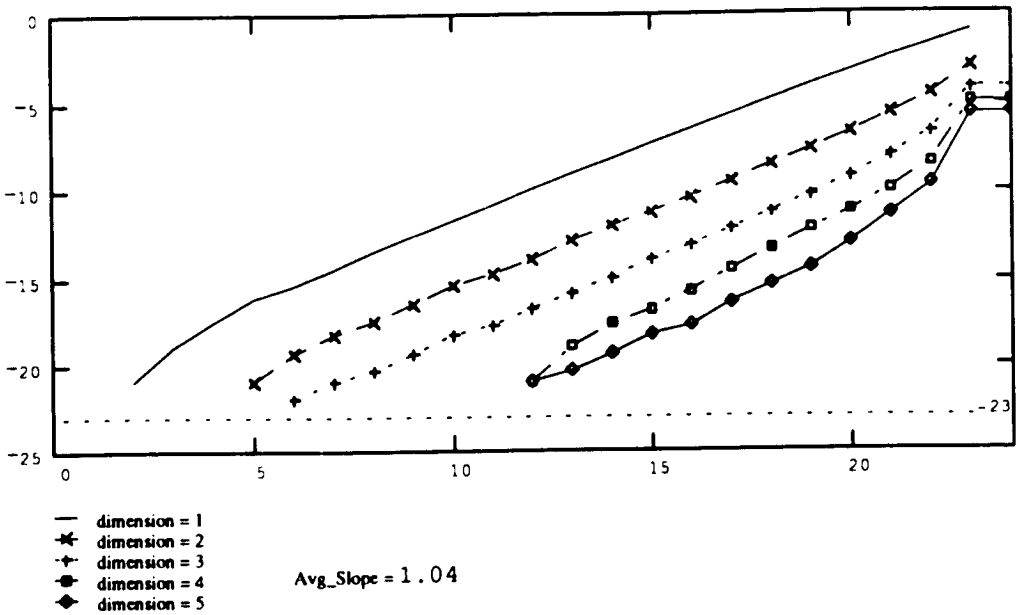


Figure 5.17 - CD plot of quadratic map with noise every 10 terms: $\log_2 C_N(r)$ vs. $\log_2(r)$.

Additional treatments of these questions from various points of view can be found in [Geweke, 1989], [Kostelich, 1988], and

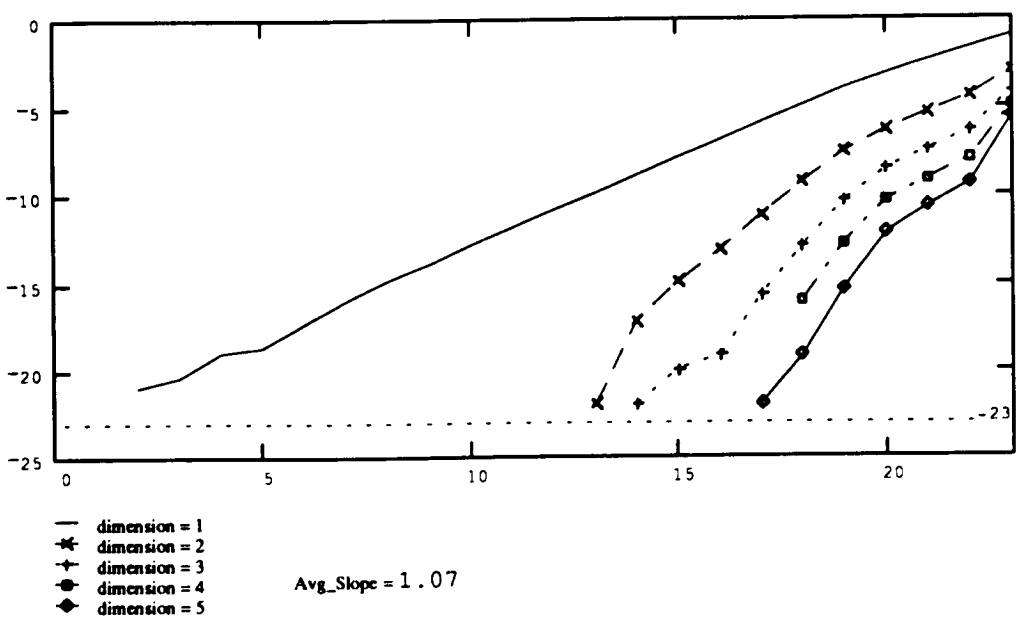


Figure 5.18 - CD plot of quadratic map with noise superimposed on data: $\log_2 C_N(r)$ vs. $\log_2(r)$.

[Tong, 1990].

5.6.2 Experimental Data from COMSIM

As described in section 4.1, by varying the parameters period (T) and phase angle (Θ), the COMSIM shell determines whether the mesh runs continuously or saturates, given a set of mesh conditions. Typically, queue length, the number of originating processors and the number of packets originated are held constant. Moreover, the number and location of any unavailable processors is fixed before data is collected.

Figure 5.19 is the 3-dimensional portrait of run-time for

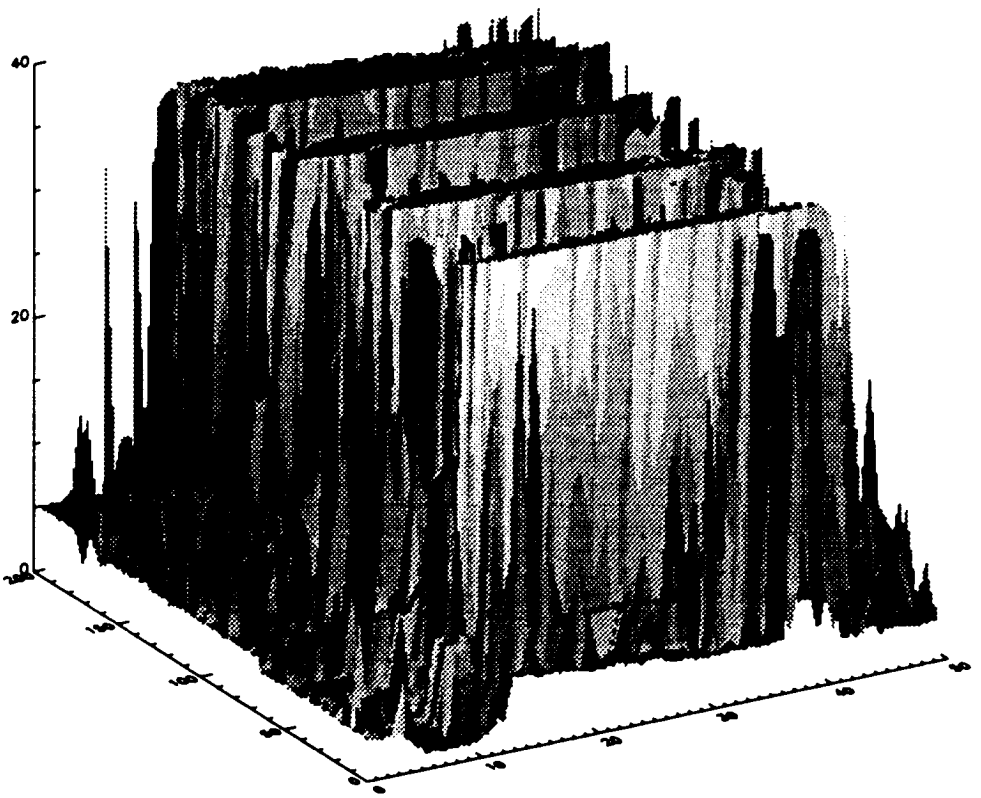


Figure 5.19 - COMSIM behavioral surface.

a 5 X 5 mesh of processors. This surface was created with the following parameter set: processor (1,1) originated one packet to (5,5) every time unit while (5,5) originated 20 packets to (1,1) based on the combination of T and Θ; processors (1,3), (3,3) and (5,3) were held unavailable (see Chapter 4 for an explanation of the processor numbering scheme); the queue length of non-originators was fixed at 5; with, T_1 = 1 and Θ_1 = 0, Θ_2 was varied from 0 to 49 time units. Phase is plotted from left to right in the figure. T_2 was varied from 10 to 800 time units. In the figure T_2 ranges from 225 to 424. Period is plotted from foreground to background. The behavior shown is typical of mid-range-load mesh operation. The figure represents 10,000 combinations of T_2 and Θ_2.

The z-axis (100's of time units) indicates that the minimum running time is 100 time units. At 4096 time units, the flat "mesa" surfaces represent large areas of free-running behavior. Interspersed peaks indicate that small zones of free-running behavior exist throughout the surface. Data points less than 4096 time units indicate that the mesh has "frozen." A frozen mesh is defined in Chapter 1 as a global deadlock condition.

The sharp transitions from saturation to ideal equilibrium, as exhibited by the mesas, are typical behavior for the 5 X 5 mesh. Figure 5.20 shows a second example of this behavior. In this case, five originators were used to generate the behavioral surface. All processors were available for packet transfer. (1,3) originated 20 packets to

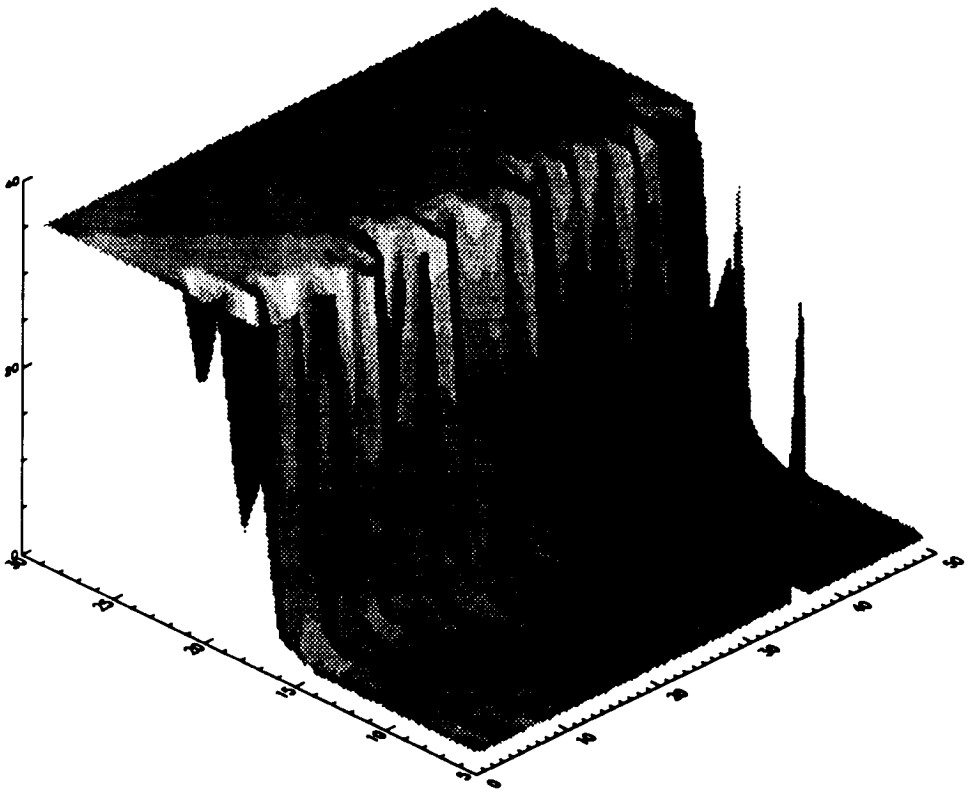


Figure 5.20 - COMSIM: Five Originators.

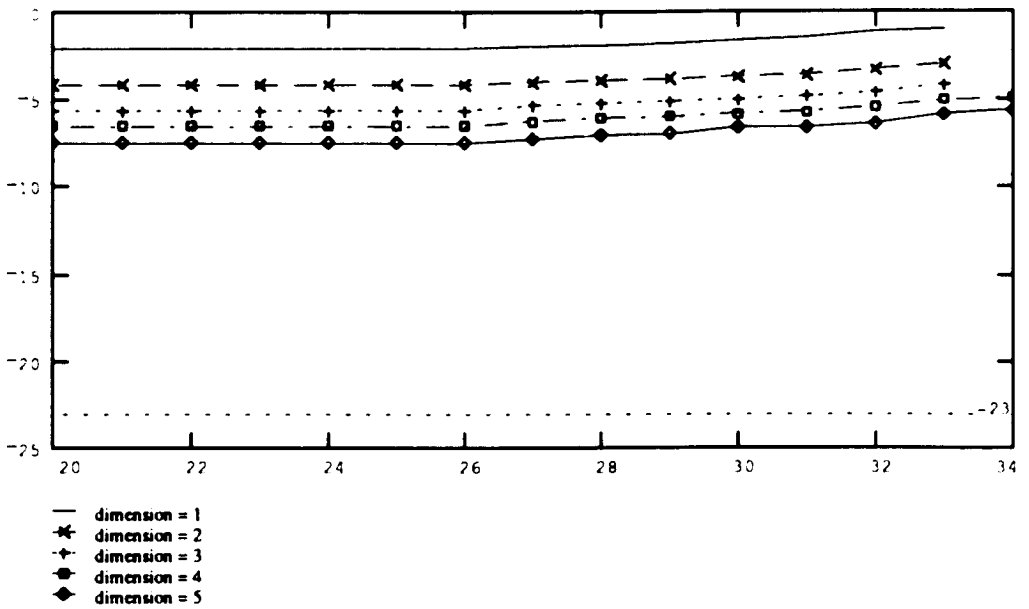
(5,3) every 50 time units (T_1 = 50 and Θ_1 = 0). The other four originators were (1,1), (1,5), (5,1) and (5,5) (i.e., the four corners of the mesh). These processors originated 20 packets to the processor diagonally across the mesh. Thus, the packets from (1,1) went to (5,5), etc. The refresh rate and phase angle of these processors were controlled by T_2 and Θ_2. The limits for these parameters were: $5 \leq T_2 \leq 85$ and $0 \leq \Theta_2 \leq 49$. The queue length for non-originators was set at 5.

The surfaces shown in Figures 5.18 and 5.19 define the critical regions of the mesh. The transition from saturation to fault tolerant conditions for the five originator mesh

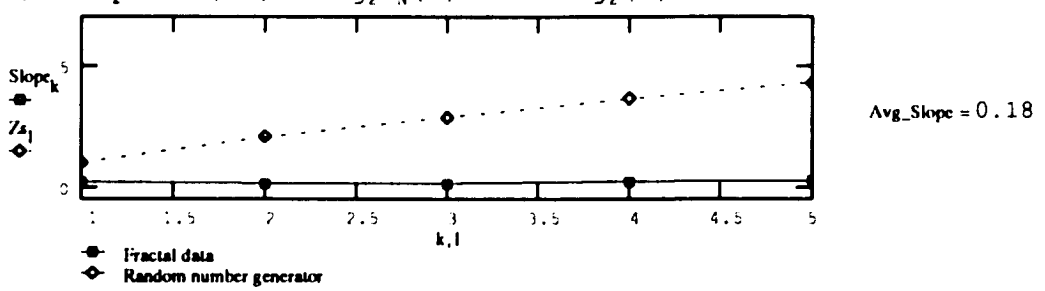
(Figure 5.20) was used to search for indications of chaotic behavior. Specifically, mesh operation was evaluated at T_2 = 16, 18 and 20, with Θ_2 = 5 though 7. T_2 = 50 and Θ_2 = 0 were used to establish an ideal-equilibrium baseline behavior for the multi-originator case. In addition, the mesh was run with one originator (1,1), originating 20 packets to (5,5), every 50 time units. This represents the linear behavioral region of the mesh.

The figures on the following pages show the progression of the mesh from light loading to heaviest loading. For example, Figure 5.20 is for T_2 = 50 (light loading), which is on the top of the mesa. Figures 5.24 is for T_2 = 16 (heavy loading), at the base of the slope.

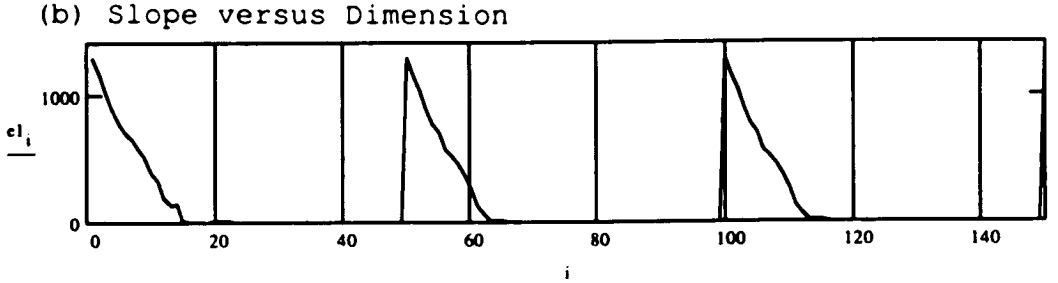
The energy of the four corner processors (E1) is used to represent the behavior of the system. Note that other energy measures exhibit similar behavior (a fractal CD). A CD plot of the energy of the mesh (EMesh), excluding the originators, demonstrates this. In each case, the CD plot, part (a), is accompanied by a plot of "Slope versus Dimension" (SVD), as part (b). The SVD plot compares the fractal part of the CD plot to random noise and specifies the CD plot's slope. Moreover, a sample of the temporal data (1st 150 time units) is included, as part (c) and linear and semi-log plots of the FFT's for this data as parts (d) and (e). A summary of the data is given at the end of this section.



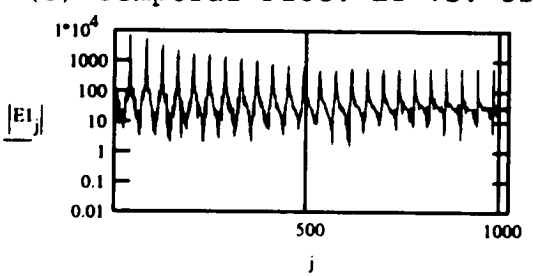
(a) CD plot (E1): $\log_2 C_N(r)$ vs. $\log_2(r)$



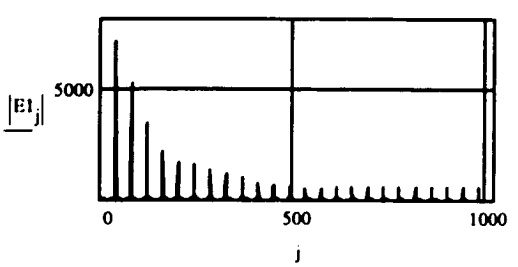
(b) Slope versus Dimension



(c) Temporal Plot: E1 vs. time

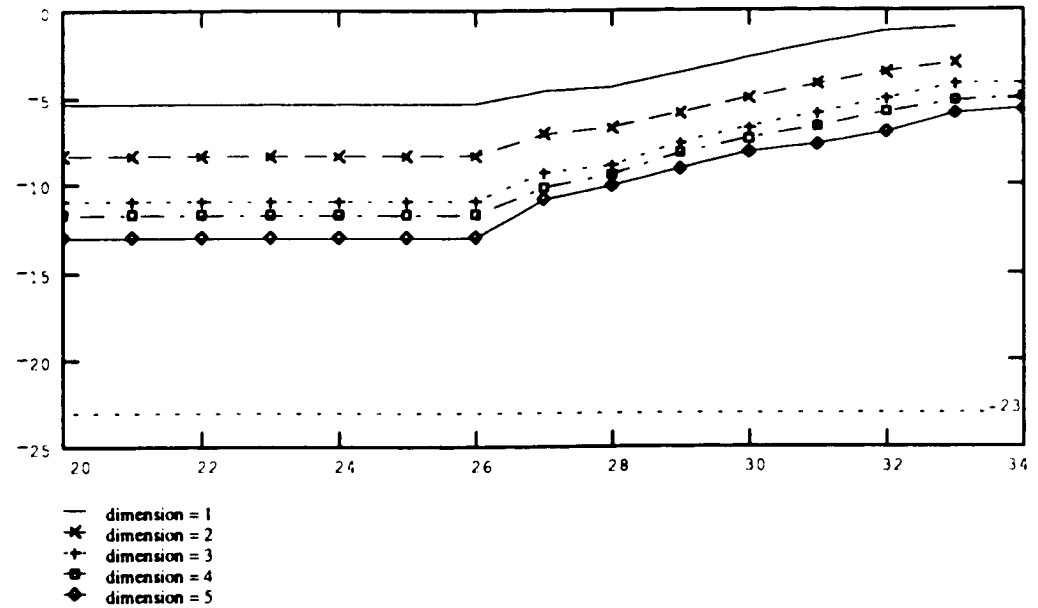


(d) FFT (E1): log scale

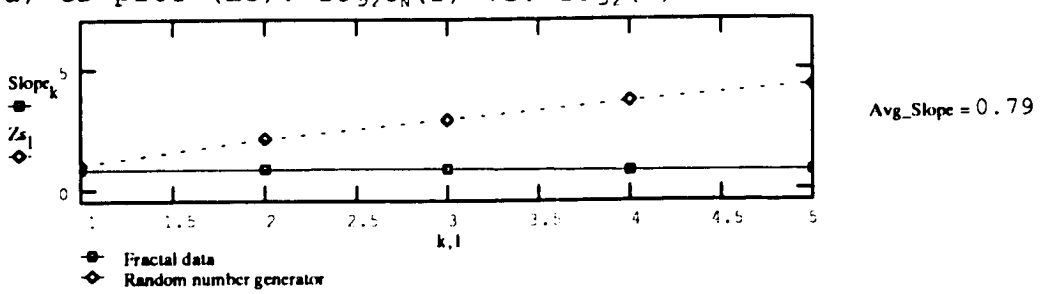


(e) FFT (E1): linear scale

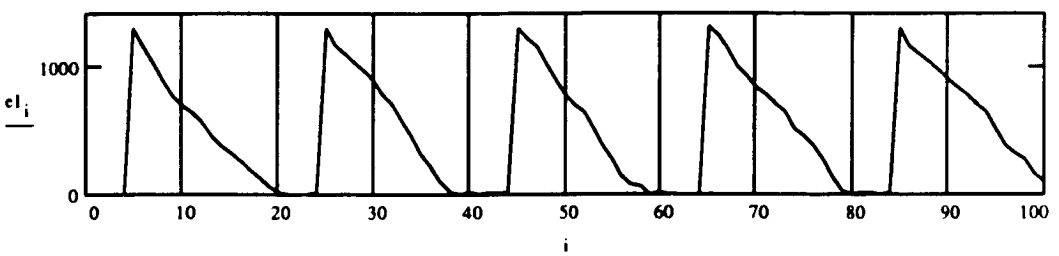
Figure 5.21 - COMSIM data: T_2 = 50, Θ_2 = 0.



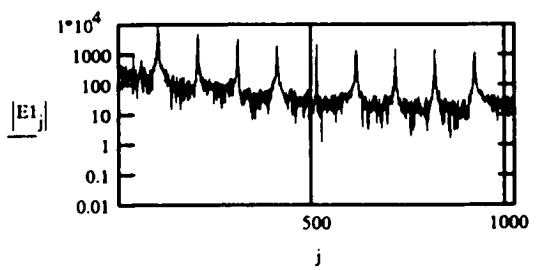
(a) CD plot (E1): $\log_2 C_N(r)$ vs. $\log_2(r)$



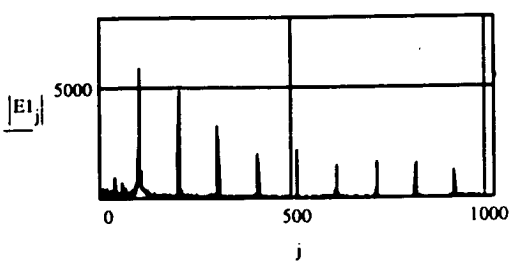
(b) Slope versus Dimension



(c) Temporal Plot: E1 vs. time

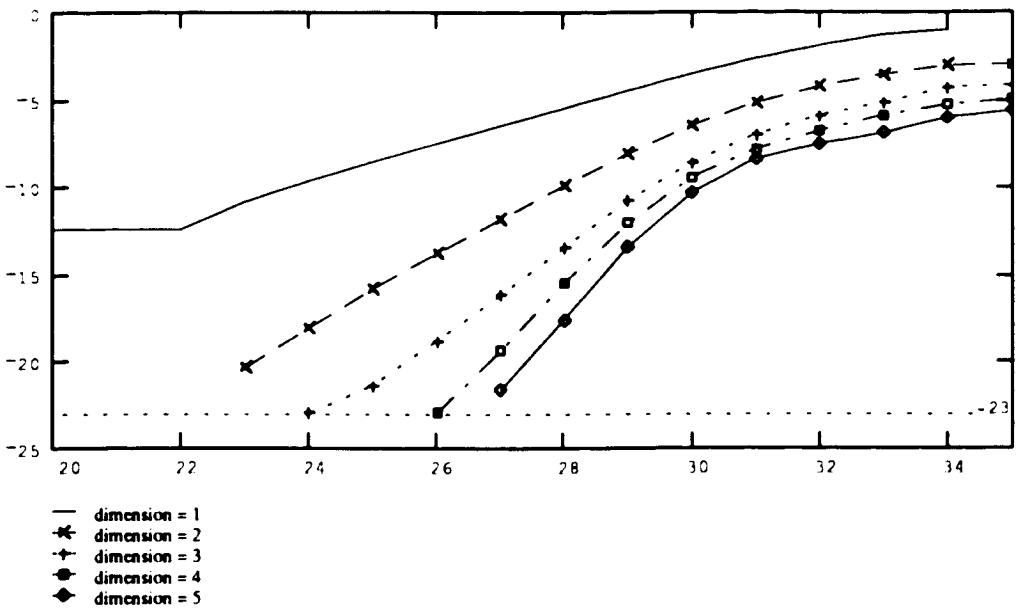


(d) FFT (E1): log scale

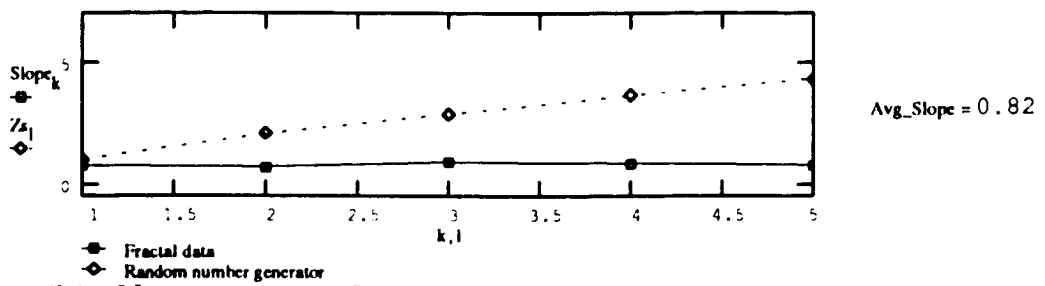


(e) FFT (E1): linear scale

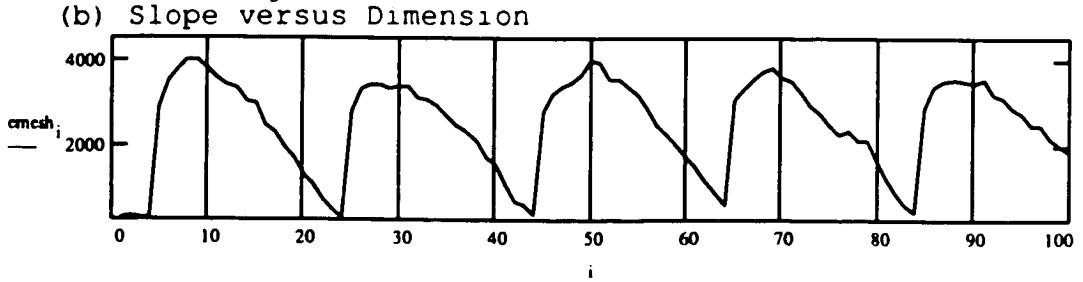
Figure 5.22 - COMSIM data: T_2 = 20, Θ_2 = 5.



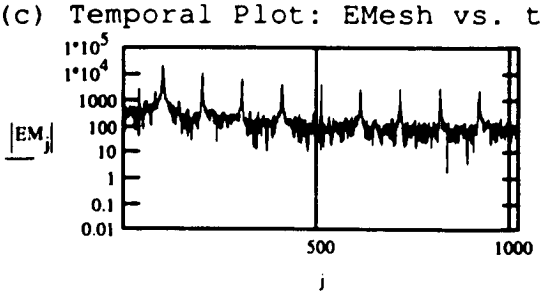
(a) CD plot (EMesh): $\log_2 C_N(r)$ vs. $\log_2(r)$



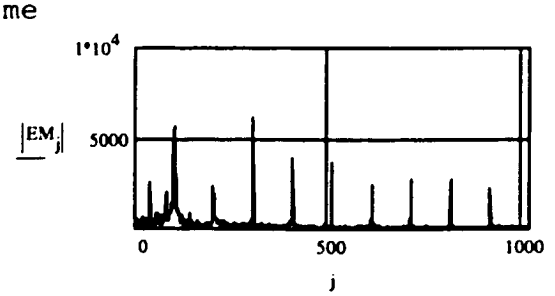
(b) Slope versus Dimension



(c) Temporal Plot: EMesh vs. time

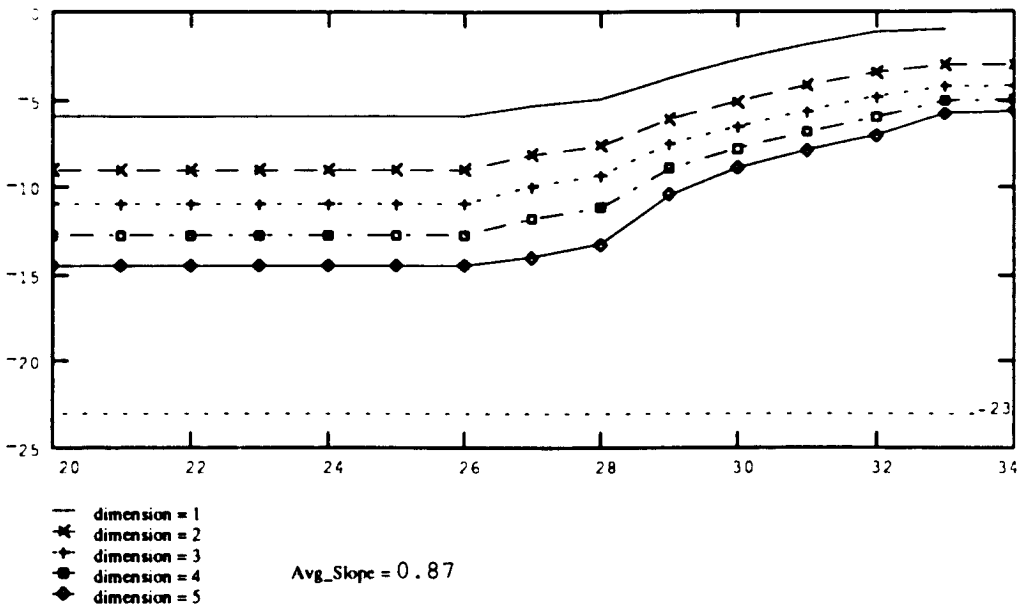


(d) FFT (EMesh): log scale

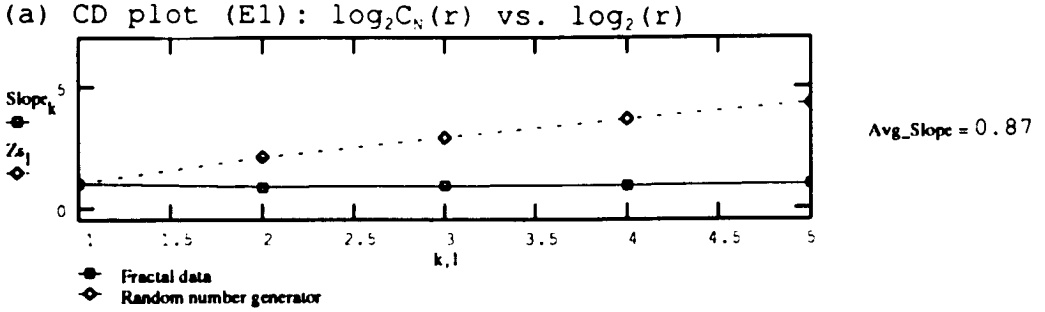


(e) FFT (EMesh): linear scale

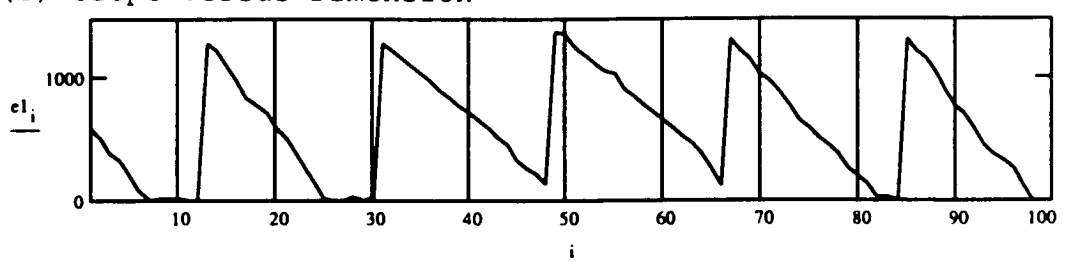
Figure 5.23 - COMSIM data: $T_2 = 20$, $\Theta_2 = 5$.



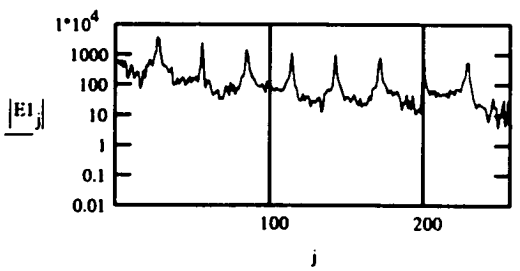
(a) CD plot (E1): $\log_2 C_N(r)$ vs. $\log_2(r)$



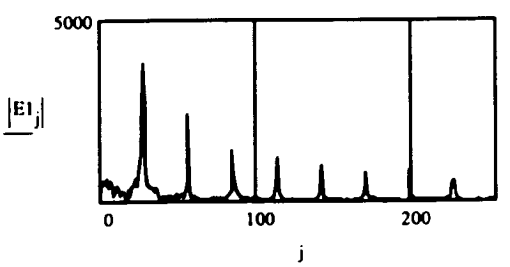
(b) Slope versus Dimension



(c) Temporal Plot: E1 vs. time

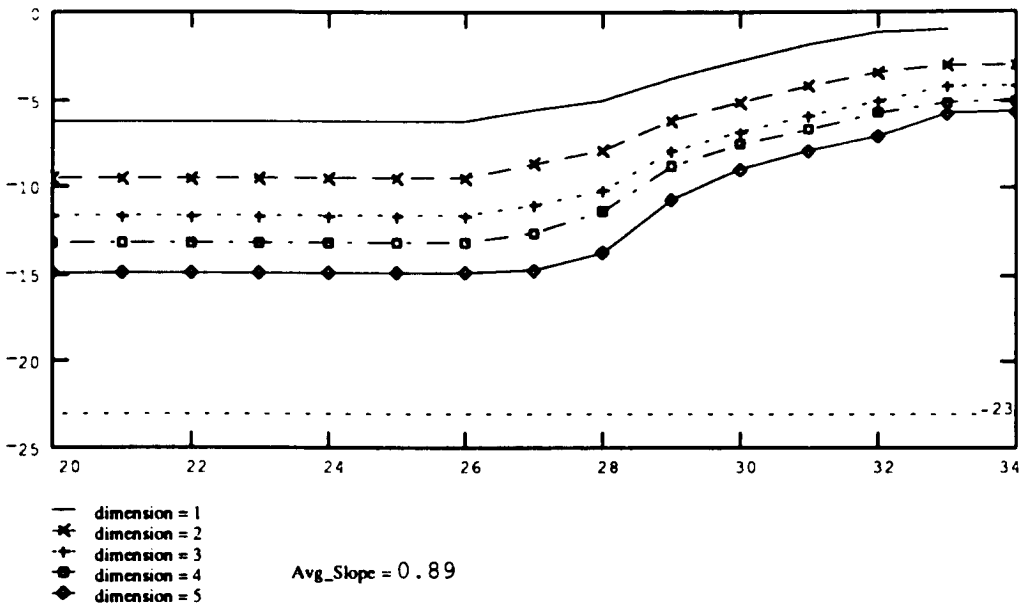


(d) FFT (E1): log scale

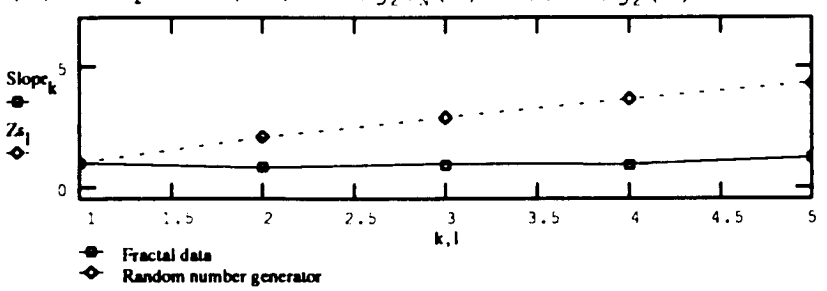


(e) FFT (E1): linear scale

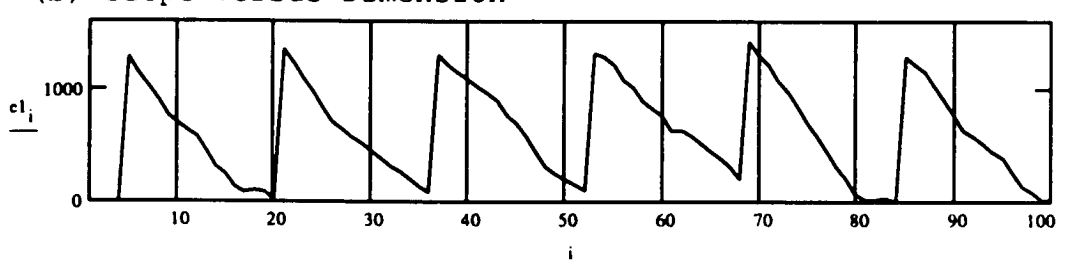
Figure 5.24 - COMSIM data: $T_2 = 18$, $\Theta_2 = 5$.



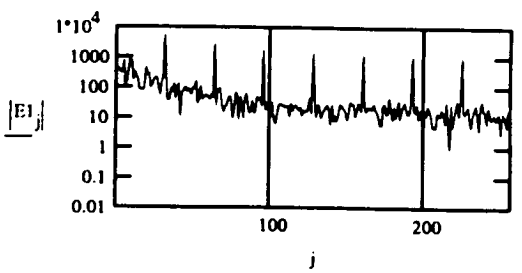
(a) CD plot (E1): $\log_2 C_N(r)$ vs. $\log_2(r)$



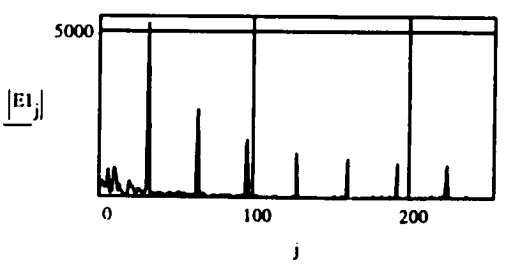
(b) Slope versus Dimension



(c) Temporal Plot: E1 vs. time

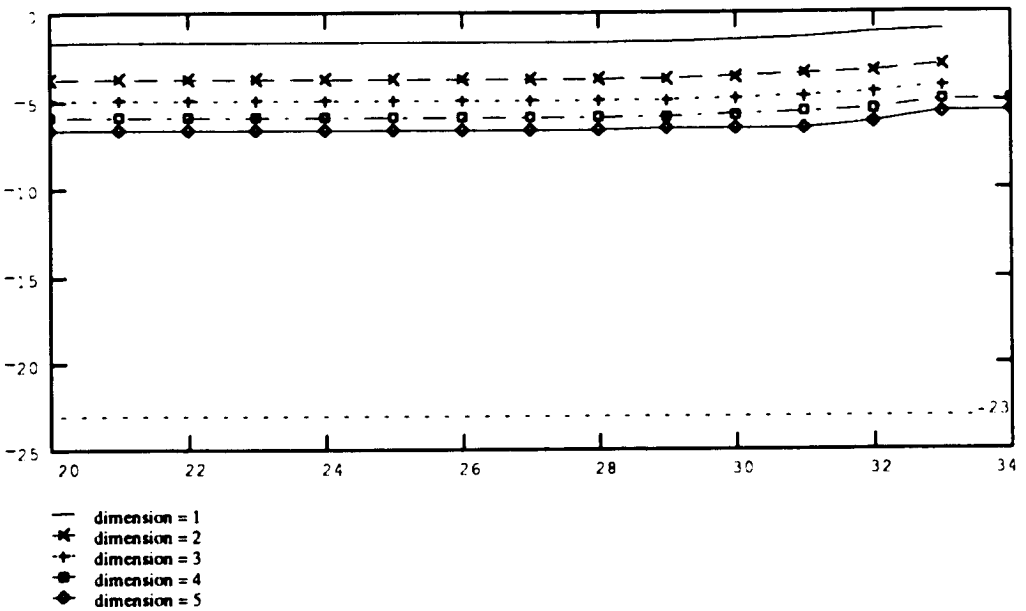


(d) FFT (E1): log scale

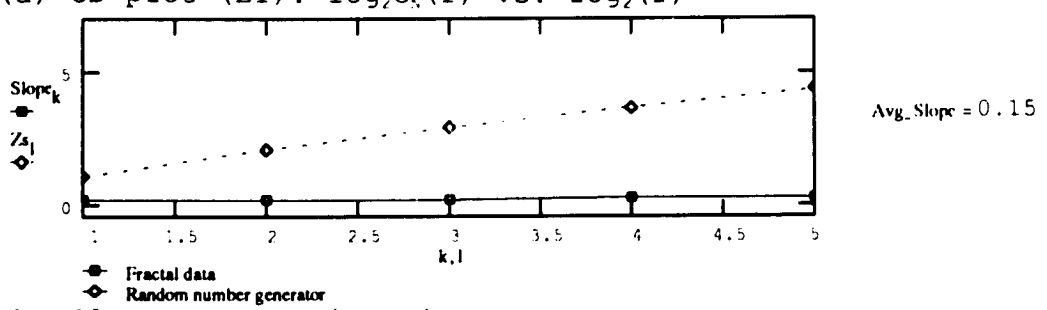


(e) FFT (E1): linear scale

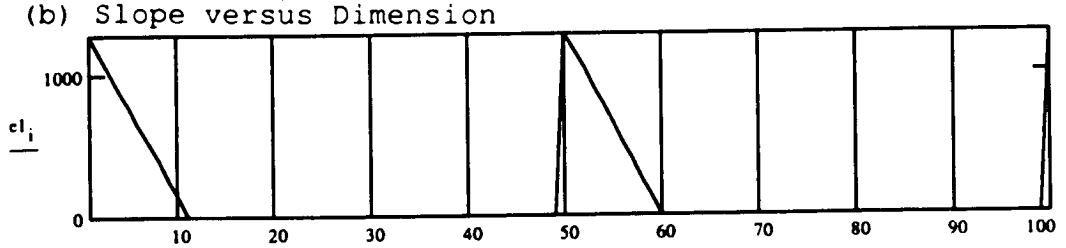
Figure 5.25 - COMSIM data: $T_2 = 16$, $\Theta_2 = 5$.



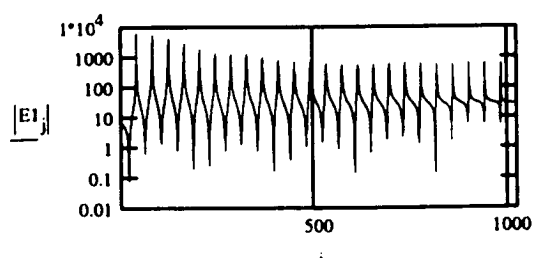
(a) CD plot (E1): $\log_2 C_k(r)$ vs. $\log_2(r)$



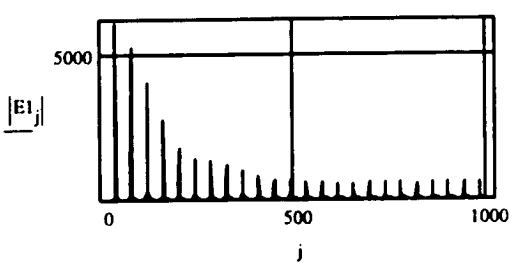
(b) Slope versus Dimension



(c) Temporal Plot: E1 vs. time



(d) FFT (E1): log scale



(e) FFT (E1): linear scale

Figure 5.26 - COMSIM data: $T_2 = 50$, $\Theta_2 = 0$ (Single Originator).

The calculation of the slope of the CD plots was made using the right-hand portion of each embedding dimension, where the slope remained constant over four or five data points. Linear regression was used to determine the slope of this portion of the curve. An average of the slopes of embedding dimensions 2, 3 and 4 was used to represent the slope of the graph. Multiple data runs yielded similar results. The CD varied approximately 3 percent of its value. Note that the CD plots for lowest mesh activity are the smoothest because the most data points were capturable when the mesh was free-running.

The FFT plots for light to heavy loads in the mesh are very similar. The shape of the linear FFT plots is indicative of a strong, periodic, triangular forcing function. "Noise" appears at the low end of the linear FFT plot for T_2 = 20. It appears to intensify as the load on the mesh increases. The semi-log version of the FFT plots are more difficult to read. Even at low load, the spectrum has a broad-band characteristic. The shape of the heavy-load FFT's is somewhat similar to the 1/f shape that [Musha, 1976] and others use as an indication of chaos. As expected, the FFT indicates that something "interesting" may be happening in the mesh. However, the results of the FFT measurements are inconclusive.

Figure 5.27 summarizes all of the correlation dimension measurements. The graph includes both data shown in Figures 5.21 through 5.26 and additional data points taken at the same period values. The three traces are:

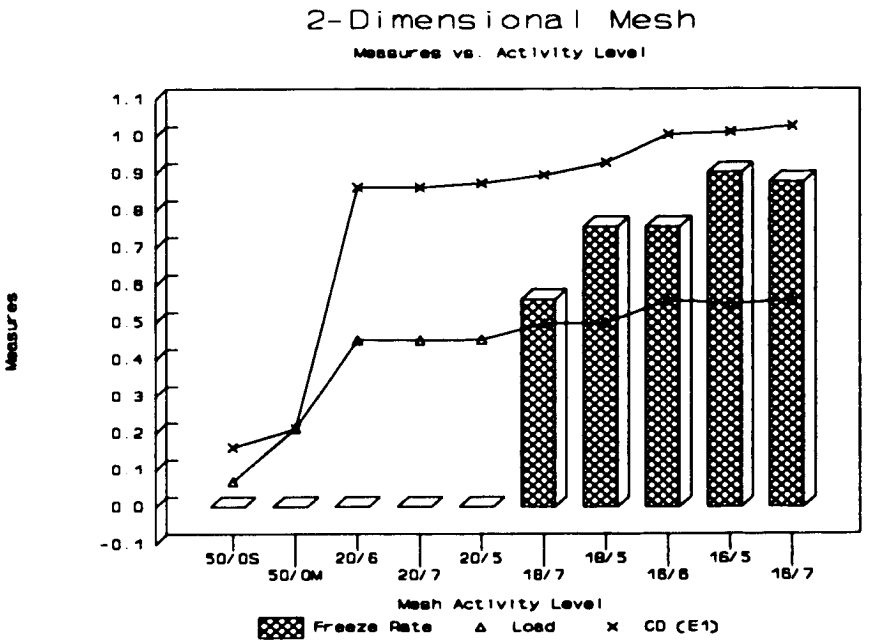


Figure 5.27 - COMSIM Results Summary.

1. Correlation dimension of the energy of the corner processors (E1).
2. Mesh loading (i.e., fraction of total communication ports occupied).
3. Freeze Rate (i.e., a measure of how readily the mesh freezes).

Data is sorted by correlation dimension.

As demonstrated by the Cantor set (Section 5.4) and quadratic map (Section 5.5.1) examples, a non-integral dimension is a clear indication of fractal behavior. The 5 X 5 mesh exhibits a fractal dimension from very light loads (T_2 = 50, Θ_2 = 0, Single Originator) through heavy loads (T_2 = 16, Θ_2 = 7, Multiple Originators). This is shown by the non-integral slope in Figures 5.21 to 5.26.

Moreover, it is clear that the correlation dimension measurement tracks system load. Thus, it is reasonable to expect that the CD can be used as a measure of the level of system activity. Note that load is a system-level measure and correlation dimension is a relatively local one. As E1 has the greatest effect on system behavior, it is reasonable to expect that the local measure may track the apparent system-level behavior. In addition, the correlation dimension of the energy of the single originator (E2) also appears to track the load for the lower half of the activity range. Further experiments are needed to determine how this CD is linked to system-level activity.

Lastly, Figure 5.23 shows what may be the combination of a fractal dimension, and "random" behavior. A modified version of the Slope versus Dimension graph is shown in Figure 5.28. The fractal component is extracted from the right-hand

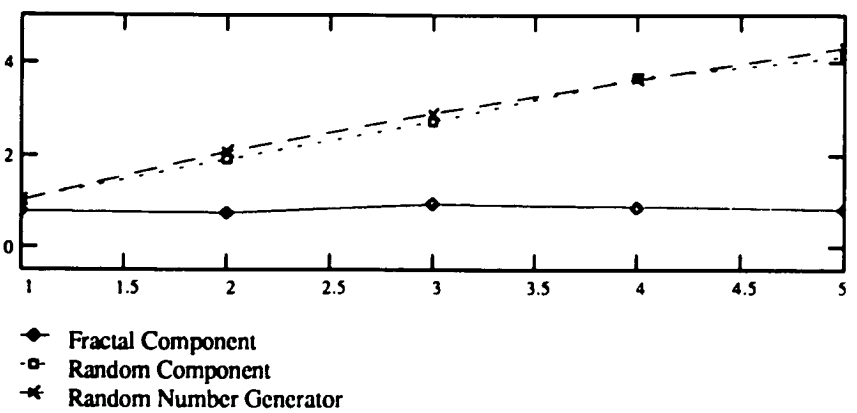


Figure 5.28 - Two Element Slope versus Dimension Plot
EMesh: T_2 = 20, Θ_2 = 5

portion of the graph. The random component is extracted from the left-hand side. The change in CD of the random component is virtually identical to that of the pseudo-random number

generator in Section 5.4. This fact and the results shown in Section 5.5.1 lead to the observation that the random component may well be the indication of random behavior in the mesh. This characteristic is also evident, to a lesser extent, in the CD plots of E1 and E2. However, the change in slope does not mirror the pseudo-random number generator.

The origin of the random behavior may be the coin-flip used to influence routing. When the mesh is under heavy load, the stochastic coin-flip is used more often to determine routing strategy. Thus, it may be possible in some sense to measure the effectiveness of the routing algorithm by examining the amount of randomness present in the CD plot. Increased randomness may be a sign that the mesh will soon freeze.

The results presented in this section clearly support the hypotheses stated in Chapter 1. To summarize:

HYPOTHESIS 1: Computing networks may be viewed as pseudo-physical systems.

HYPOTHESIS 2: The combination of correlation dimension and fast fourier transform may be used as the measure of activity level in pseudo-physical systems.

CHAPTER VI

IMPLICATIONS OF THE RESEARCH

6.1 Applications and Impact

This dissertation introduces a new computing network paradigm based on system-level thinking. This paradigm includes quantitative and qualitative aspects. As described in Chapter 2, the ultimate goal of this research is to use the pseudo-physical computing network paradigm to design computing networks with distributed, dynamic routing control and congestion avoidance. Therefore, these become promising areas for prototyping the new metrics developed in Chapter 5.

In packet switched networks, routing control is used to select either an end-to-end path for virtual circuits or the next hop for datagram routing (see Chapter 2). Congestion avoidance deals with the rate at which packets are allowed to enter the network. The goals of both include efficient movement of packets from message source to message sink while avoiding situations in which the network would freeze. Each uses measures of network performance, such as load and delivery delay, to make decisions (see Chapter 2). In order to operate efficiently, each requires recent, system-level information about the network. System-level information can be approximated by the behavior of a significant portion of the network surrounding the processor making the decision. It is reasonable to believe that the metrics introduced in

Chapter 5 would be a valuable asset in making these decisions.

6.1.1 Dynamic Routing Example

[Kumar, 1992] discusses a distributed, dynamic, virtual circuit routing algorithm based on a locally calculated parameter. This parameter, K, is used to select the best available path. K is defined as

$$K \propto \frac{C\beta}{NT} \tag{6.1}$$

Where: C = capacity of lowest-capacity link

β = reliability of the path

N = number of hops in the path

T = total waiting time (i.e., round-trip response time)

The authors have included additional factors to improve upon the power criterion proposed in [Kleinrock, 1976] and discussed in [Mitra, 1992]. A promising area for future work is the use of the correlation dimension and the FFT measurements to improve this approach.

6.1.2 Congestion Avoidance Example

Feedback flow control techniques [Rose, 1992], [Shenker, 1990] use one or more feedback bits to indicate local congestion to sources of packet traffic. In feedback flow control, the source responds to the feedback bits by reducing the amount of traffic offered to the network. Another scheme uses tokens to allow packets access to the network [Sidi, 1993]. Moreover, the probability of packet loss has been suggested as a mechanism for controlling packet transmission

rates [Williamson, 1991]. It may be possible to use the Chapter 5 metrics to detect local congestion.

6.2 Future Directions

A "chaos-friendly" computing network would use real-time correlation dimension (CD) and fast Fourier transform (FFT) measurements at each node to route packets and adjust packet flow rates. Real-time means that spectral calculations and CD slope approximations are made while the network is running. A real-time CD tool does not exist at present. Given what has been accomplished thus far, future work will include at least the following elements:

1. metric qualification,
2. dynamic measurement,
3. system control, and
4. real system / task experiments.

The results shown in section 5.5.2 indicate that the CD of the energy of a single originator (E_2) is influenced by the large energy changes occurring in the rest of the system. To some extent this CD may be viewed as a system-level measure of activity.

A number of intriguing questions arise. For example, given a set of network conditions, what does the CD of a non-originator indicate about the activity level of that processor? What does it indicate about nearby processors? To what extent are the CD measurements "clustered" (i.e., an indication of different behavioral regions within the mesh).

It will be interesting to see how the CD's of processors, at varying distances from the traffic flow, are affected by the large movement of packets.

A possible approach to answering these questions includes the following experiments:

1. Measure the CD of each node in the 25 processor mesh under varying mesh conditions.
    a) Load the mesh using one, two and five originators with varying topologies.
    b) Vary T_2 and Θ_2 to produce linear behavior, fault-tolerant behavior, etc.
2. Correlate the CD data for "clusters" of nodes (i.e., look for communities)
3. Repeat steps 1 and 2 for a 400 node mesh.

Experiments with larger meshes will allow for a wider range of experimental dynamics. In other words, a finer grained measure of activity will be possible. We suspect that some behaviors, that are analogous to that of some form of ecological structure, will be noticeable in a 400 processor mesh.

Two things need to be done to use the CD as a real-time measurement tool:

1. determine what constitutes a reasonable "working set" of time-series data, and
2. design a CD plot evaluation mechanism.

The idea of a working set comes from memory management techniques used in operating systems [Silbershatz, 1991]. The

working set constitutes an overlapping of measurements in time. The number of points that must be taken to approximate the CD is unclear. [Brock, 1986] indicates that 200 data points may be enough. [Grassberger, 1983] uses greater than 20,000 measurements for some CD plots. However, 2048 data points gave good results in Chapter 5. Moreover, the amount of allowable overlapping needs to be determined.

Neural networks are a promising tool for recognizing and evaluating the fractal, and possibly random, part of the CD plot. Multi-stage neural networks have been used to recognize kanji characters and to scan time-series data for uniquely shaped spikes [Hammerstrom, 1993]. Each node might use a multi-stage neural network to determine an appropriate region in which to evaluate fractal dimension and to provide the dimension itself.

The control of "chaotic" systems is a relatively new area of research [Ditto, 1993]. Recently [Petrov, 1993], demonstrated a control mechanism for the oscillatory Belousov-Zhabotinsky (BZ) reaction. The flow rate of chemical reactants into a stirred-tank reactor determines whether the reaction is steady-state, periodic or chaotic. Using the approach proposed by [Ott, 1990], Petrov et al. stabilize the chaotic reaction by confining it to a periodic orbit. As a result, period-1 and period-2 behaviors were selectable.

A fundamental goal of this research is to utilize the global chaotic dynamics of a computing network. Note that chaos in the BZ reaction is a global behavior. Viewing the

system globally while controlling congestion locally is an ideal situation. The combination of the chaos-control ideas and the Chapter 5 time-series metrics is an especially exciting area for future work.

Lastly, and perhaps most importantly, once the characteristics of the metrics are better understood, they should be applied to real computing networks and used to characterize the multiprocessor solution of numerical problems. We hope to experiment with a massively parallel computer system (e.g., an Intel multiprocessor system) to determine its CD/FFT profile. It will be interesting to see how the profile changes while the processors cooperate to solve real problems. Note that COMSIM may be used to simulate the cooperation of a mesh of processors cooperating in some of these computations.

CHAPTER VII

CONCLUSIONS

The combination of local computing capability with packet switched communication invites the creation of a new entity - the computing network. Currently, load-delay congestion is a limiting factor in computing networks. Because computing networks are pseudo-physical systems, their complex behavior emulates the behavior of many natural systems. This is indicated by the presence of a fractal dimension combined with a rich spectral signature. It is possible for many natural systems to operate close to their functional capacity. Combining the ideas of computing networks and pseudo-physical systems creates a new paradigm. This paradigm captures the behavior of computing networks for the purpose of improved communication and processing capability.

The hypotheses of this research were:

1: Computing networks may be viewed as pseudo-physical systems.

2: The combination of fractal dimension and spectral signature may be used to characterize the behavior of a pseudo-physical system.

7.1 Systems Analysis

To investigate the hypotheses, a formal Systems Analysis, including a taxonomy of system models, was undertaken. The

following areas of competence are integrated into this Engineering Systems Design dissertation: system models, mathematical interpretation of artificial systems, system visualization, chaos, applications and team management. At the conclusion of the Systems Analysis, a methodology for the remaining work was adopted.

A four-node analysis methodology, for the two-dimensional mesh of processors, or any physical system, was developed. this methodology consists of a functional description, probabilistic model, simulation and behavioral model. This methodology forms a high-level guide for the remainder of the dissertation research.

Three tools for analysis, in the form of two deterministic, discrete-time, discrete-space simulations and one stochastic, discrete-time, continuous space model, were developed and characterized.

7.2 DDD Simulations

The flexibility of a simulator for modeling the two-dimensional mesh of processors was shown. Moreover, the COMSIM simulator can be used in the future to model different types of networks. (The proprietary DECSIM router is of similar merit.)

As a result of the substantial amount of data collected, it was firmly established that the fast Fourier transform, even in combination with the X(t) versus X(t+1) attractor plots, is not sufficient to prove whether chaos exists within

the mesh, nor is it sufficient to provide behavioral signatures for critical operating regions. Tantalizing hints of interesting behavior lurk in the data, but the FFT alone is insufficient to demonstrate it.

7.3 Network Dynamics

In this research only the combination of the correlation dimension and the fast Fourier transform was able to capture the behavioral model of the two-dimensional mesh of processors. Other metrics, such as the Poincare map and bifurcation diagram were inadequate.

The experimental results, shown in Chapter 5, clearly support the two hypotheses within the context of our test environment. Computing networks, as represented by the two-dimensional mesh of processors, appear to exhibit a low-dimensional chaotic behavior. Thus, computing networks may be viewed as pseudo-physical systems. Moreover, the correlation dimension, in conjunction with the fast Fourier transform, may be used as a dynamic, "global" measure of the level of system activity.

The potential applications and impact of the research clearly point to an exciting future. Combining chaotic dynamics with computing networks promises to unlock their unused capability.

REFERENCES

[Ahluwalia, 1992] A.K. Ahluwalia and M. Singhal, Performance Analysis of the Communication Architecture of the Connection Machine, *IEEE Transactions on Parallel and Distributed Systems*, vol. 3, no. 6, November 1992, pp. 728-738.

[Antola, 1990] A. Antola and N. Scarabottolo, Reconfiguration of FFT Arrays: A Flow-Driven Approach, *Proceedings of the International Conference on Application Specific Array Processors*, Princeton, NJ, September 1990, pp. 401-413.

[Axelrod, 1984] R. Axelrod, *The Evolution of Cooperation*, Basic Books, New York, 1984.

[Baker, 1990] G.L. Baker and J.P. Gollub, *Chaotic Dynamics: An Introduction*, Cambridge University Press, Cambridge, 1990.

[Bedrosian, 1987] S.D. Bedrosian and D.L. Jaggard, A Fractal Graph Approach to Large Networks, *Proceedings of the IEEE*, vol. 75, no. 7, July 1987, pp. 966-968.

[Ben-Mizrachi, 1984] A. Ben-Mizrachi, I. Procaccia and P. Grassberger, Characterization of Experimental (Noisy) Strange Attractors, *Physics Review A*, vol. 29, no. 3, pp. 975-977.

[Berge, 1984] P. Berge et al., *Order within Chaos*, John Wiley and Sons, New York, 1984.

[Berliner, 1991] L.M. Berliner, Likelihood and Bayesian Prediction of Chaotic Systems, *Journal of the American Statistics Association*, vol. 86, 1991, pp. 938-952.

[Berliner, 1992] L.M. Berliner, Statistics, Probability and Chaos, *Statistical Science*, vol. 7, no. 1, January 1992, pp. 69-90.

[Bertsekas, 1992] D. Bertsekas and R. Gallager, *Data Networks*, Prentice Hall, Englewood Cliffs, NJ, 1992.

[Blaine, 1993] L.G. Blaine and P.G. Drexel, An Experimental Study of Communication in a Datagram Mesh Simulator, *Proceedings of the 1993 International Simulation Technology Multiconference*, San Francisco, CA, November 1993, to appear.

[Blaine, 1991] L. Blaine, Theory vs. Computation in Some Very Simple Dynamical Systems, *The College Mathematics Journal*, vol. 22, no. 1, January 1991, pp. 42-44.

[Boghosian, 1990] B.M. Boghosian, Computational Physics on the Connection Machine, *Computers in Physics*, January / February 1990, pp. 14-33.

[Brock, 1986] W.A. Brock, Distinguishing Random and Deterministic Systems: Abridged Version, *Journal of Economic Theory*, vol. 40, 1986, pp. 168-195.

[Callens, 1985] P. Callens, Congestion Control System for Fast Data Networks, *IBM Technical Disclosure Bulletin*, vol. 27, no. 8, January 1985, pp. 4976-4977.

[Chang, 1992] S. Chang, Fair Integration of Routing and Flow Control in Communication Networks, *IEEE Transactions on Communications*, vol. 40, no. 4, April 1992, pp. 821-834.

[Chatterjee, 1992] S. Chatterjee and M.R. Yilmaz, Chaos, Fractals and Statistics, *Statistical Science*, vol. 7, no. 1, January 1992, pp. 49-68.

[Christie, 1993] P. Christie, A Fractal Analysis of Interconnection Complexity, *Proceedings of the IEEE*, Pre-print.

[Crutchfield, 1980] J. Crutchfield et al., Power Spectral Analysis of a Dynamical System, *Physics Letters*, Vol. 76A, No. 1, March 1980, pp. 1-4.

[Danet, 1976] A. Danet et al., The French Public Packet Switching Service: The TRANSPAC

Network, *Proceedings of the 3rd ICCC*, Toronto, ON, August 1976, pp. 251-260.

[Devaney, 1989] R.L. Devaney, *Introduction to Chaotic Dynamical Systems*, 2nd edition, Benjamin Cummings, Menlo Park, CA, 1989.

[Ditto, 1993] W.L. Ditto and L.M. Pecora, Mastering Chaos, *Scientific American*, vol. 269, no. 2, August 1993, pp. 78-84.

[Drazin, 1992] P.G. Drazin, *Nonlinear Systems*, Cambridge University Press, Cambridge, 1992.

[Drexel, 1992] P.G. Drexel, A. Rucinski and Z. Shen, A Probabilistic Model of a Mesh System with Distributed Routing, *Proceedings of the Pacific-Rim International Conference on Modelling, Simulation and Identification*, Vancouver, BC, August, 1992, pp. 39-42.

[Drexel, 1993] P.G. Drexel, Z. Shen and L. Urbach, Characterization and Implementation of a Mathematical Model of a Mesh System with Dynamic Routing, *Proceedings of the 1993 Simulation Multiconference on the High Performance Computing Symposium*, Arlington, VA, March, 1993, pp. 176-181.

[Eberhardt, 1990] S. Eberhardt, Dedicated Processor for Funding Lowest-Cost Map Path, *JPL Invention Report NPO-17716/7219*, May 1990.

[El Mesbahi, 1990] J. El Mesbahi, Nearest Neighbor Problems on a Mesh-Connected Computer, *IEEE Transactions on Man and Cybernetics*, vol. 20, no. 5, September-October 1990, pp. 1199-1204.

[Evans, 1988] J.B. Evans, *Structures of Discrete Event Simulation: An Introduction to the Engagement Strategy*, John Wiley and Sons, New York, 1988.

[Feigenbaum, 1979] M.J. Feigenbaum, The Onset Spectrum of Turbulence, *Physics Letters*, Vol. 74A, No. 6, December 1979, pp. 375-378.

[Fenstermacher, 1979] P.R. Fenstermacher and H.L. Swinney, Bifurcations to Periodic, Quasiperiodic, and Chaotic Regimes in Rotating and Convecting Fluids, *Bifurcation Theory and Applications in Scientific Disciplines*, O. Gurel and O.E. Rhossler, editors, New York Academy of Sciences, New York, 1979, pp. 652-666.

[FitzGerald, 1993] J. FitzGerald, *Business Data Communications*, John Wiley and Sons, New York, 1993.

[Garcia-Lunes-Aceves, 1993] J.J. Garcia-Lunes-Aceves, Loop-Free Routing Using Diffusing Computations, *IEEE/ACM Transactions on Networking*, vol. 1, no. 1, February 1993, pp. 130-141.

[Garzia, 1990] M.R. Garzia, Discrete Event Simulation Methodologies and Formalisms, *Simulation Digest*, vol. 21, no.1, Summer 1990, pp. 3-13.

[Gaughan, 1993] P.T. Gaughan and S. Yalamanchili, Adaptive Routing Protocols for Hypercube Interconnection Networks, *IEEE Computer*, vol. 26, no. 5, May 1993, pp. 12-23.

[Gaver, 1973] D.P. Gaver, and G.L. Thompson, *Programming and Probability Models in Operations Research*, Brooks Cole, Monterey, CA, 1973.

[Gershenfeld, 1988] N. Gershenfeld, An Experimentalist's Introduction to the Observation of Dynamical Systems, *Directions in Chaos, (Vol. 2)*, World Scientific, New Jersey, 1988.

[Geweke, 1989] J. Geweke, Interference and Forcasting for Deterministic Non-linear Time Series Observed with Measurement Errors *Non-linear Dynamics and Evolutionary Economics*, R. Day and P. Chen, editors, 1989, To appear.

[Gibson, 1991] G.A. Gibson, *Computer Systems*, Prentice Hall, Englewood Cliffs, NJ, 1991.

[Gleick, 1987] J. Gleick, *Chaos: Making a New Science*, Viking Press, New York, 1987.

[Goldberger, 1990] A.L. Goldberger and D.R. Rigney, Sudden Death Is Not Chaos, *The Ubiquity of Chaos*, S. Krasner editor, American Association for the Advancement of Science, Washington, 1990, pp. 23-34.

[Gollub, 1980] J.P. Gollub and S.V. Benson, Many Routes to Turbulent Convection, *Journal of Fluid Mechanics*, Vol. 100, Part 3, 1980, pp. 449-470.

[Graham, 1989] L.R. Graham, D.E. Knuth and O. Patashnik, *Concrete Mathematics*, Addison-Wesley, Reading, MA, 1989.

[Grassberger, 1983] P. Grassberger and I. Procaccia, Measuring the Strangeness of Strange Attractors, *Physica 9D*, 1983, pp. 189-208.

[Grinstein, 1990] G. Grinstein et al., Perception-driven Data Visualization, Technical Report, Computer Science Department, University of Lowell, January 1990.

[Guckenheimer, 1992] J. Guckenheimer and P. Holmes, *Nonlinear Oscillations, Dynamical Systems and Bifurcations of Vector Fields*, Springer-Verlag, New York, 1983.

[Gulick, 1992] D. Gulick, *Encounters with Chaos*, McGraw-Hill, Inc., New York, 1992.

[Habib, 1992] I.W. Habib and T.N. Saadawi, Access Flow Control Algorithms in Broadband Networks, *Computer Communications*, vol. 15, no. 5, June 1992, pp. 281-332.

[Hac, 1991] A. Hac, Congestion Control and Switch Buffer Allocation in High-Speed Networks, *Proceedings of IEEE Infocom '91*, Bal Harbour, FL, April 1991, pp. 314-322.

[Haken, 1981] H. Haken, Chaos and Order in Nature, *Chaos and Order in Nature*, H. Haken editor, Springer Verlag, Heidelberg, 1981, pp. 2-11.

[Halsall, 1992] F. Halsall, *Data Communications, Computer Networks, and Open Systems*, Addison-Wesley, Wokingham, England, 1992.

[Hammerstrom, 1993] D. Hammerstrom, Neural Networks at Work, *IEEE Spectrum*, vol. 30, no. 6, June 1993, pp. 26-32.

[Handley, 1993] J.W. Handley, et al., Chaos and Fractal Algorithms Applied to Signal Processing and Analysis, *Simulation*, April 1993, pp. 261-278.

[Hayt, 1967] W.H. Hayt, *Engineering Electromagnetics*, McGraw-Hill, New York, 1967.

[Huberman, 1988] B.A. Huberman and T. Hogg, The Behavior of Computational Ecologies, *The Ecology of Computation*, B.A. Huberman, editor, Elsevier Science Publishing Company, New York, 1988.

[Iansiti, 1985] M. Iansiti, et al., Noise and Chaos in a Fractal Basin Boundary Regime of a Josephson Junction, *Physical Review Letters*, Vol. 55, No. 7, August 1985, pp. 746-749.

[Ilyadis, 1990] N. Ilyadis, The Simulation of the Non-linear Dynamic Behavior of Distributed Routing Networks using DECSIM, *Master's Thesis*, University of New Hampshire, December 1990.

[Ilyadis, 1991] N. Ilyadis, P. Drexel and A. Rucinski, The Simulation of the Non-linear Dynamic Behavior of Distributed Routing Networks using DECSIM, *Proceedings of 1991 Winter Simulation Conference*, Phoenix, AZ, December 1991, pp. 705-715.

[Inselberg, 1989] A. Inselberg and B. Dimsdale, Visualizing Multi-variate Relations with Parallel Coordinates, *Proceedings of the 3rd International Conference on Human-Computer Interaction*, Boston, MA, September 1989, pp. 460-467.

[Kim, 1988] C. Kim and D.A. Reed, Adaptive Packet Routing in a Hypercube, *Third Conference on Hypercube Concurrent Computers and Applications*, vol. 1, Pasadena, CA, January 1988, pp. 625-630.

[Kleinrock, 1976] L. Kleinrock, *Queueing Systems, Volume II: Computer Applications*, John Wiley and Sons, New York, 1976.

[Kobb, 1993] B.Z. Kobb, Personal Wireless, *IEEE Spectrum*, vol. 30, no. 6, June 1993, pp. 20-25.

[Kostelich, 1988] E.J. Kostelich and J.A. Yorke, Noise Reduction in Dynamical Systems, *Physics Review A*, vol. 38, pp. 1649-1652.

[Kumar, 1992] A. Kumar and S. Singh, Simulation Based Performance Study of a Dynamic Routing Algorithm, *Simulation Digest*, vol. 22, no. 1, Summer 1992, pp. 28-38.

[Lam, 1987] S.S. Lam and C. Hsieh, Modeling, Analysis and Optimal Routing of Flow-Controlled Communication Networks, *Technical Report TR-87-24*, University of Texas at Austin, June 1987.

[Law, 1991] A.M. Law, *Simulation Modeling and Analysis*, McGraw-Hill, New York, 1991.

[Lee, 1992] T.H. Lee, Design and Analysis of a New Self-Routing Network, *IEEE Transactions on Communications*, vol. 40, no. 1, January 1992, pp. 171-177.

[Licker, 1987] P.S. Licker, *Fundamentals of Systems Analysis*, Boyd and Fraser, Boston, MA, 1987.

[Lim, 1992] S.L. Lim and R.E. Newman-Wolfe, Relationships Between Network Parameters and the Performance of Distributed Adaptive Routing Algorithms Using Learning Automata in Packet-Switched Datagram Networks, *Proceedings: Applications of Artificial Intelligence X: Knowledge-Based Systems*, SPIE vol. 1707, Orlando, FL, April 1992, pp. 117-125.

[Lorenz, 1963] E.N. Lorenz, Deterministic Non-periodic Flow, *Journal of Atmospheric Science*, vol. 20, 1963, pp. 130-141.

[Mandelbrot, 1977] B.B. Mandelbrot, *The Fractal Geometry of Nature*, W.H. Freeman and Co., New York, 1977.

[Martin, 1990] J. Martin, *Telecommunications and the Computer*, Prentice Hall, Englewood Cliffs, NJ, 1990.

[Martland, 1989] D. Martland, Dynamic Behaviour of Boolean Networks, *Neural Computing Architectures*, Ed. Igor Aleksander, The MIT Press, 1989.

[McClelland, 1986] J.L. McClelland et al., The Appeal of Parallel Distributed Processing, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, D.E. Rumelhart and J.L. McClelland editors, MIT Press, Cambridge, MA, 1986.

[McKenney, 1991] P.E. McKenney, Stochastic Fairness Queuing, *Internetworking: Research and Experience*, vol. 2, no. 2, June 1991, pp. 113-131.

[Miller, 1988] M.S. Miller and K.E. Drexler, Comparitive Ecology: A Computational Perspective, *The Ecology of Computation*, B.A. Huberman, editor, Elsevier Science Publishing Company, New York, 1988.

[Mitra, 1992] D. Mitra, Asymptotically Optimal Design of Congestion Control for High Speed Data Networks, *IEEE Transactions on Communications*, vol. 40, no. 2, February 1992.

[Moon, 1992] F.C. Moon, *Chaotic and Fractal Dynamics*, John Wiley and Sons, New York, 1992.

[Moreno, 1990] J.H. Moreno and T. Lang, Matrix Computations on Systolic-Type Meshes, *IEEE Computer*, vol. 23, no. 4, April 1990, pp. 32-50.

[Morrison, 1991] F. Morrison, *The Art of Modeling Dynamic Systems*, John Wiley and Sons, New York, 1991.

[Musha, 1976] T. Musha and H. Higuchi, The 1/f Fluctuation of a Traffic Current on an Expressway, *Japanese Journal of Applied Physics*, Vol. 15, No. 7, July 1976, pp. 1271-1275.

[Ni, 1993] L.M. Ni and P.K. McKinley, A Survey of Wormhole Routing Techniques in Direct Networks, *IEEE Computer*, vol. 26, no. 2, February 1993, pp. 62-76.

[Nickolls, 1991] J.R. Nickolls, A New Technology: The Massively Parallel Computer System, *ASIC Technology and News*, April 1991, pp. 30-33.

[Octavio, 1984] M. Octavio and C. Readi Nasser, Chaos in a DC-bias Josephson Junction in the Presence of Microwave Radiation, *Physical Review B*, Vol. 30, No. 3, August 1984, pp. 1586-1588.

[Onvural, 1990] R.O. Onvural, Survey of Closed Queueing Networks with Blocking, *ACM Computing Surveys*, vol. 22, no. 2, June 1990, pp. 83-121.

[Ott, 1990] E. Ott, C. Grebogi and J.A. Yorke, *Physical Review Letters*, vol. 64, March 12, 1990, pp. 1196-1199.

[Parker, 1987] T.S. Parker and L.O. Chua, Chaos: A Tutorial for Engineers, *Proceedings of the IEEE*, vol. 75, no. 8, August 1987, pp. 982-1008.

[Passamante, 1989] A. Passamante and P.E. Rapp, Measures of Dimensions from Astrophysical Data, *Measures of Complexity and Chaos*, Plenum Press, New York, 1989.

[Patterson, 1993] D.A. Patterson and J.L. Hennessy, *Computer Organization and Design*, Morgan Kaufmann, San Mateo, CA, 1993.

[Petrov, 1993] V. Petrov, et al., Controlling Chaos in the Belousov-Zhabotinsky Reaction, *Nature*, vol. 361, January 21, 1993, pp. 240-243.

[Poincare, 1913] H. Poincare, *The Foundation of Science: Science and Method*, published 1913, English Translation, The Science Press, Lancaster, PA, 1946.

[Pollacia, 1989] L.F. Pollacia, A Survey of Discrete Event Simulation and State-of-the-Art Discrete Event Languages, *Simulation Digest*, vol. 20, no. 3, Fall 1989, pp. 8-25.

[Price, 1978] W.L. Price, Some Comments on Simulated Datagram Store-and-Forward Networks, *Computer Networks*, vol. 2, no. 1, February 1978, pp. 70-73.

[Price, 1978] W.L. Price, Simulation Studies of Data Communication Networks Operating in Datagram Mode, *Computer Journal*, vol. 21, no. 3, August 1978, pp. 219-223.

[Pritsker, 1986] A.A.B. Pritsker, *Introduction to Simulation and SLAM II*, John Wiley and Sons, New York, 1986.

[Raatikainen, 1992] K.E.E. Raatikainen, Modeling Service Distributions in Network Simulation, *Simulation*, vol. 59, no. 2, August 1992, pp. 116-126.

[Roberts, 1970] L. Roberts and B. Wessler, Computer Network Development to Achieve Resource Sharing, *Proceedings of the Spring Joint Computer Conference*, 1970.

[Rose, 1992] O. Rose, The Q-bit Scheme: Congestion Avoidance using Rate Adaptation, *Computer Communication Review*, vol. 22, no. 2, April 1992, pp. 29-42.

[Rucinski, 1990] A. Rucinski, P. Drexel and B. Dziurla, Chaotic Nature of Mesh Networks with Distributed Routing, *Proceedings of International Symposium on Advances in Interconnection and Packaging*, Boston, MA, November 1990, pp. 388-398.

[Rucinski, 1991] A. Rucinski, P. Drexel and B. Dziurla, An Integrated Model of Complex Behavior in Networks with Distributed Routing, *Frontiers of Computing Systems Research*, Vol. III, S.K. Tewksbury, editor, Plenum Press, New York, 1991, pp.291-316.

[Ruelle, 1990] D. Ruelle, Deterministic Chaos: The Science and the Fiction, *Proceedings of the Royal Society of London*, vol. A 427, February 1990, pp. 241-248.

[Sayers, 1990] C.L. Sayers, Chaos and the Business Cycle, *The Ubiquity of Chaos*, S. Krasner editor, American Association for the Advancement of Science, Washington, 1990, pp. 115-125.

[Schroeder, 1991] M. Schroeder, *Fractals, Chaos, Power Laws: Minutes from an Infinite Paradise*, W.H. Freeman and Co., New York, 1991.

[Schwartz, 1980] M. Schwartz and T.E. Stern,Routing Techniques Used in Computer Communication Networks, *IEEE Transactions on Communications*, vol. COM-28, no. 4, April 1980, pp. 539-552.

[Schwartz, 1988] M. Schwartz, *Telecommunications Networks: Protocols, Modeling and Analysis*, Addison-Wesley, Reading,MA, 1988.

[Seeley, 1989] T.D. Seeley, *The Honey Bee Colony as a Superorganism*, American Scientist, vol. 77, November-December 1989, pp. 546-553.

[Shaw, 1981] R. Shaw, Modeling Chaotic Systems, *Chaos and Order in Nature*, H. Haken editor, Springer Verlag, Heidelberg, 1981, pp. 218-231.

[Shen, 1993] Z. Shen, P. Drexel and L. Urbach, Packet Delay Prediction in Datagram Mesh Systems, *Proceedings of the 1993 ACM/SIGAPP Symposium on Applied Computing*, Indianapolis, IN, February, 1993, pp. 539-545.

[Shenker, 1990] S. Shenker, A Theoretical Analysis of Feedback Flow Control, *Computer Communication Review*, vol. 20, no. 4, September 1990, pp. 156-165.

[Shivaratri, 1992] N.G. Shivaratri et al., Load Distributing for Locally Distributed Systems, *IEEE Computer*, vol. 25, no. 12, December 1992, pp. 33-44.

[Sidi, 1993] M. Sidi et al., Congestion Control Through Input Rate Regulation, *IEEE Transactions on Communications*,vol. 41, no. 3, March 1993, pp. 471-477.

[Silbershatz, 1991] A. Silbershatz, J.L Peterson and P.B. Galvin, *Operating System Concepts*, Addison-Wesley, Reading, MA, 1991.

[Smith, 1982] J.M. Smith, *Evolution and the Theory of Games*, Cambridge University Press, Cambridge, 1982.

[Smith, 1989] M.J. Smith and G. Salvendy, Work with Computers: Organizational, Management, Stress, and Health Aspects, *Proceedings of the Third International Conference on Human-Computer Interaction, Vol 1*, Elsevier Science Publishing Company, New York, 1989.

[Stallings, 1991] W. Stallings, *Data and Computer Communications*, Macmillan, New York, 1991.

[Stallings, 1993] W. Stallings, *Local and Metropolitan Area Networks*, Macmillan, New York, 1993.

[Stoll, 1990] C. Stoll, *The Cuckoo's Egg*, Simon and Schuster, New York, 1990.

[Stone, 1990] H.S. Stone, *High-Performance Computer Architecture*, Addison-Wesley, Reading, MA, 1990.

[Stout, 1962] M.B. Stout, *Basic Electrical Measurements*, Prentice Hall, Englewood Cliffs, NJ, 1962.

[Sushchenko, 1989] S.P. Sushchenko, Analysis of End-to-End Message Delay in a Multilink Virtual Circuit, *Automatika i Vychislitel'naya Technika*, vol. 23, no. 3, 1989, pp. 48-58.

[Taam, 1989] R.E. Taam and B.A. Fryxell, The Hydrodynamica of Accretion from Stellar Winds, *American Scientist*, vol. 77, November-December 1989, pp. 539-545.

[Takens, 1980] F. Takens, Detecting Strange Attractors in Turbulance, *Dynamical Systems and Turbulence*, D. Rand and L. Young, editors, Warwick, 1980, pp. 366-382.

[Takens, 1983] F. Takens, Distinguishing Deterministic and Random Systems, *Nonlinear Dynamics and Turbulence*, G. Borenblatt, G. Iooss and D. Joseph, editors, Pitman, Boston, MA, pp. 314-333.

[Tanenbaum, 1989] A.S. Tanenbaum, *Computer Networks*, Prentice Hall, Englewood Cliffs, NJ, 1989.

[Tanenbaum, 1992] A.S. Tanenbaum et al., Parallel Programming Using Shared Objects and Broadcasting, *IEEE Computer*, vol. 25, no. 8, August 1992, pp. 10-19.

[Tewksbury, 1987] S.K. Tewksbury et al., Distributed Routing Algorithm for Mesh Networks: The Impact of Execution Times and Time Uncertainties, Technical Report, AT&T Bell Labs, 1987.

[Tewksbury, 1988] S.K. Tewksbury, Communication Network Issues and High-Density Interconnects in Large-Scale Distributed Computing Systems, *IEEE Journal on Selected Areas in Communications*, vol. 6, no. 3, April 1988, pp. 587-609.

[Tong, 1990] H. Tong, *Non-linear Time Series: A Dynamical System Approach*, Oxford University Press, 1990.

[Tredicce, 1988] J.R. Tredicce and N.B. Abraham, Experimental Measurements to Identify and/or Characterize Chaotic Signals, *Lasers and Quantum Optics*, CIF Series Vol. 13, World Scientific, New Jersey, 1988.

[Trivedi, 1982] K.S. Trivedi, *Probability and Statistics with Reliability, Queuing and Computer Science Applications*, Prentice Hall, Englewood Cliffs, NJ, 1982.

[Tufillaro, 1992] N.B. Tufillaro, et al., *An Experimental Approach to Nonlinear Dynamics and Chaos*, Addison-Wesley, Reading, MA, 1992, pp. 166-183.

[Tymes, 1971] L. Tymes, TYMNET-A Terminal Oriented Communication Network, 1971 Spring Joint Computing Conference, *AFIPS Conference Proceedings*, vol. 38, 1971, pp. 211-216.

[Voldman, 1981] J. Voldman and L.W. Hoevel, The Software-Cache Connection, *IBM Journal of Research and Development*, Vol. 25, No. 6, November 1981, pp. 877-893.

[Voldman, 1983] J. Voldman and L.W. Hoevel, Fractal Nature of Software-Cache Interaction, *IBM Journal of Research and Development*, Vol. 27, No. 2, March 1983, pp. 164-170.

[Wang, 1992] Z. Wang and J. Crowcroft, Eliminating Periodic Packet Losses in the 4.3-Tahoe BSD TCP Congestion Control Algorithm, *Computer Communication Review*, vol. 22, no. 2, April 1992, pp. 9-16.

[Weems, 1992] C.C. Weems et al., Image Understanding Architecture: Exploiting Potential Parallelism in Machine Vision, *IEEE Computer*, vol. 25, no. 2, February 1992, pp. 65-68.

[Wetherbe, 1984] J.C. Wetherbe, *Systems Analysis and Design*, West, St. Paul, MN, 1984.

[Wheeler, 1991] K.R. Wheeler, Chaotic Behavior in a Recurrent Neural Network, *Technical Report ECE.ISG.91.01*, University of New Hampshire, January 1991.

[Whitten, 1989] J.L. Whitten, L.D. Bentley and V.M. Barlow, *Systems Analysis and Design Methods*, Irwin, Boston, MA, 1989.

[Williamson, 1991] C.L. Williamson and D.R. Cheriton, Load-Loss Curves: Support for Rate-Based Congestion Control in High-Speed Datagram Networks, *Computer Communication Review*, vol. 21, no. 4, September 1991, pp. 17-28.

[Zorpette, 1991] G. Zorpette, Minis and Mainframes, *IEEE Spectrum*, vol. 28, no. 1, January 1991, pp. 40-43.

APPENDIX

In the four lemmas that follow, we assume that a real, bounded time series $X = x_1, x_2, \ldots$ has correlation dimension υ.

LEMMA: $\upsilon \leq 1$.

PROOF: Since we assume boundedness, we may as well assume that $x_k \in [0,1)$ for all k. Now let N and m be positive integers, and let $\varepsilon = 1/m$. For $i = 1, 2, \ldots, m$, define N_i to be the number of terms of X_N in the interval $[(i-1)/m , i/m)$. Our immediate goal is to show that

$$\varepsilon < C_N(\varepsilon) \tag{A.1}$$

Since the terms of X_N in any given subinterval differ by at most ε, it follows that

$$\frac{\left(\sum_{i=1}^{m} N_i^2\right)}{N^2} \leq C_N(\varepsilon) \tag{A.2}$$

so it is sufficient to show that

$$N^2 \leq m\sum_{i=1}^{m} N_i^2 \tag{A.3}$$

i.e.

$$\left[\sum_{i=1}^{m} N_i\right]^2 \leq m\sum_{i=1}^{m} N_i^2 \tag{A.4}$$

Now recall the Cauchy-Schwarz inequality, which says that for vectors A and B in R^n ,we have $(A\cdot B)^2 < (A\cdot A)(B\cdot B)$. (Here $\cdot$ is the ordinary dot product.) Apply this to A = (1,1, ... ,1) and B = $(N_1, N_2, \ldots, N_m)$, and (1) follows. Now, (1) implies that $\varepsilon < C(\varepsilon)$, from which we get $\log(\varepsilon) < \log(C(\varepsilon))$. Since $\log(\varepsilon^{\upsilon}) < 0$,

$$\frac{\log(C(\varepsilon))}{\log(\varepsilon^{\upsilon})} \leq \frac{\log(\varepsilon)}{\log(\varepsilon^{\upsilon})} = \frac{1}{\upsilon} \qquad (A.5)$$

As $\varepsilon \to 0$, i.e. as $m \to \infty$, the left-hand side of the inequality above approaches 1, which is impossible unless $\upsilon \leq 1$. (We have assumed that $\upsilon \neq 0$ here, but if $\upsilon = 0$, there is nothing to prove anyway.)

NOTE: This lemma can be extended to show that the correlation dimension of a time series in R^n is at most n. Assume that the terms are all in $[0,1)^n$ and use the maximum norm. Divide the hypercube $[0,1)^n$ into r^n subcubes, with r = 1/m as before, and show that $r^n \leq C_N(r^n)$.

LEMMA: Suppose that there is a number $k > 0$ such that $C(\varepsilon) \leq k\varepsilon$, for all ε sufficiently small. then $\upsilon = 1$.

PROOF:

$$\frac{\log(C(\varepsilon))}{\log(\varepsilon)} \geq \frac{\log(k) + \log(\varepsilon)}{\log(\varepsilon)} = \frac{\log(k)}{\log(\varepsilon)} + 1 \qquad (A.6)$$

As $\varepsilon \to 0$, the left side of this inequality approaches υ, and the right side approaches 1, so $\upsilon \geq 1$. Since from the previous lemma $\upsilon \leq 1$, we have $\upsilon = 1$.

LEMMA: Suppose that there is a constant $k > 0$ such that $k \leq C(r)$, for all r. Then $\upsilon = 0$.

PROOF: From $k \leq C(r)$, we get

$$0 \leq \frac{\log(C(r))}{\log(r)} \leq \frac{\log(k)}{\log(r)} \qquad (A.7)$$

and the conclusion follows immediately from the fact that the right-hand side approaches 0 as $r \to 0$. (Note that we don't need the assumption that the CD exists here; we have shown that it is defined and is 0.)

LEMMA: Suppose that X has probability distribution φ, and suppose that φ is bounded. Then $\upsilon = 1$.

PROOF: We take the interval of definition once again to be [0,1]. Suppose that $\varphi(x) \leq M$, for $0 < x < 1$. Then

$$C(\varepsilon) = \int_0^1 \left\{ \int_{x-\varepsilon}^{x+\varepsilon} \varphi(y)\,dy \right\} \varphi(x)\,dx \leq 2M\varepsilon \int_0^1 \varphi(x)\,dx \leq 2M^2\varepsilon \qquad (A.8)$$

Now apply the previous lemma.

Next we establish the sufficient condition for the correlation dimension to exist that was mentioned earlier.

LEMMA: Suppose that $C(\varepsilon)$ is well-defined for sufficiently small ε and that for some $a > 0$ and some $\upsilon \in [0,1]$,

$$\lim_{\varepsilon \to 0} \frac{C(\varepsilon)}{\varepsilon^{\upsilon}} = a \qquad (A.9)$$

Then the correlation dimension of X is defined and is υ.

PROOF: Define: $h(\varepsilon) = (C(\varepsilon)/\varepsilon^{\upsilon}) - a$. Then $h(\varepsilon) \to 0$ as $\varepsilon \to 0$. It follows that $C(\varepsilon) = \varepsilon^{\upsilon}(a + h(\varepsilon))$, and thus $\log(C(\varepsilon)) = \upsilon\log(\varepsilon) + \log(a + h(\varepsilon))$. Dividing gives

$$\frac{\log(C(\varepsilon))}{\log(\varepsilon)} = \upsilon + \frac{\log(a + h(\varepsilon))}{\log(\varepsilon)} \qquad (A.10)$$

The second term on the right approaches 0 as ε approaches 0, and the result follows from the definition of the correlation. (Note that the conclusion would still be valid if we required only that h be bounded.)